5-2014

A Framework For Learning Scene Independent Edge Detection

Aaron J. Wilbee

A Framework For Learning Scene Independent Edge Detection

by

Aaron J. Wilbee

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science
in Electrical and Microelectronic Engineering

Supervised by

Professor Dr. Ferat Sahin
Department of Electrical and Microelectronic Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
May 2014

Approved by:

Dr. Ferat Sahin, Professor
Thesis Advisor, Department of Electrical and Microelectronic Engineering

---

Dr. Eli Saber, Professor
Committee Member, Department of Electrical and Microelectronic Engineering

---

Dr. Mark A. Hopkins, Associate Professor
Committee Member, Department of Electrical and Microelectronic Engineering

---

Dr. Sohail A. Dianat, Professor
Department Head, Department of Electrical and Microelectronic Engineering

Thesis Release Permission Form

Rochester Institute of Technology
Kate Gleason College of Engineering

Title:

A Framework For Learning Scene Independent Edge Detection

I, Aaron J. Wilbee, hereby grant permission to the Wallace Memorial Library to reproduce my thesis in whole or part.

Aaron J. Wilbee

Date

Dedication

This work is dedicated to all those who have helped make it possible. My wife, Jessica, who is ever by my side, my family who have always been there to support and inspire me, my friends who have stood by me, and my God who has watched over me.

Acknowledgments

There are many people who have made my journey through my scholastic career possible and I would like to take this opportunity to thank them.

Dr. Ferat Sahin for his constant guidance and direction through my academic journey at RIT.

Rochester Institute of Technology for providing the resources needed for me to explore my passion

My peers and fellow research assistants at the Multi-Agent Biorobotics Laboratory for all their support.

Ryan Bowen and Shitij Kumar for their aid in my undergraduate work as well as in this thesis.

Ugur Sahin for his insight and guidance in research. For the rest not mentioned I thank you all.

Abstract

A Framework For Learning Scene Independent Edge Detection

Aaron J. Wilbee

Supervising Professor: Dr. Ferat Sahin

In this work, a framework for a system which will intelligently assign an edge detection filter to an image based on features taken from the image is introduced. The framework has four parts: the learning stage, image feature extraction, training filter creation, and filter selection training. Two prototypes systems of this framework are given. The learning stage for these systems is the Berkeley Segmentation Database coupled with the Baddelay Delta Metric. Feature extraction is performed using a GIST methodology which extracts color, intensity, and orientation information. The set of image features are used as the input to a single hidden layer feed forward neural network trained using back propagation. The system trains against a set of linear cellular automata filters which are determined to best solve the *edge image* according to the Baddelay Delta Metric. One system uses cellular automata augmented with a fuzzy rule. The systems are trained and tested against the images from the Berkeley Segmentation Database. The results from the testing indicate that systems built on this framework can perform better than standard methods of edge detection on average across many types of images.

List of Contributions

Publications

- A.J. Wilbee,"A Framework for Intelligent Creation of Edge Detection Filters," submitted to *Systems, Man, and Cybernetics (SMC)*, Hong Kong, 2015.

Contributions

- Design of modular framework for diverse scene edge detection filter creation.
- Development of Fuzzy Cellular Automata implementation for framework
- Development of Linear Cellular Automata implementation for framework.
- Results verifying that edge detection filters are highly correlated to the scenes that they operate on, and that the scenes themselves are highly separable.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Image processing is a growing area of inquiry in electrical engineering with ever increasing applications in both the public and private sectors. One sub-discipline which has grown out of this field is computer vision. In computer vision, many tasks including image segmentation [6], background detection [7], and object detection [8] require that the contour of the objects of interest be found. To succeed at this task the complete edge information of the object must be detected. This thesis concerns the edge detection problem.

Edge detection is fascinating due to its nature as an ill posed problem; the true definition of an edge is unknown and the definitions used by humans are subjective [9]. Initial research in this area has yielded a number of edge detectors which have come to be the standard edge detection methods: Sobel, Canny, Prewitt, Roberts, and Laplacian of Gaussian. These detectors define an edge as a high frequency change in an image [10]. These methods, particularly Canny, yield good *edge images* in many cases which is why they are standards in many image processing tool boxes [11]. However there are many instances where this definition of an edge is incorrect. A basic example would be the fur of an animal. In many images fur will have many high frequency changes. When a person is asked what the edge of an animal is in an image, typically the animal is outlined rather than filled

in. In addition, noise in an image is also a high frequency change, but is rarely an edge. To handle these issues the standard methods have the ability to adjust parameters to specialize for the situations. The danger of specializing to a specific situation is that the same filter will then perform worse in all other situations. In addition, because images are discrete and noisy systems it is challenging to specialize these standard methods by hand to all the necessary situations. Thus, as research has progressed, efforts at tuning definition based edge detectors have shifted to machine learning approaches [11].

Rather than explicitly defining an edge, researchers have been attempting to train edge detectors, which are suited to particular types of edges and images thus allowing the computer to learn the edge definition. This is done by extracting features from the images to train against. Many approaches using features, texture gradients and spectral clustering to name a few, have been used to train these edge detectors. The features, once extracted, are used in a learning algorithm such as: neural networks, genetic algorithms, and particle swarm optimization. The learning algorithm is then used to create a new edge detector. The limitation which arises when employing these methods is their lack of universality. Though the filter, or set of filters, produced by this type of approach will work better than standard methods for a particular class of images, they are observed to behave erratically outside of that image type.

For many applications, this trade off is unimportant, the fields of: manufacturing, fault detection, and security being a few examples. For these applications the environment is typically highly controlled and known beforehand. As such, a specialized method for edge detection makes sense. For applications where the environment is not or cannot be known

beforehand, specialized edge detection begins to encounter issues. Specifically, the images which have the properties that the edge detection method are expecting, have their edge information extracted well. However, for images with different properties the same methods will perform poorly. For applications in mobile robotics and photography, this situation is unacceptable. It is for this reason that this limitation of edge detection is being investigated in this thesis.

The two current methods for edge detection, using a definition to produce an edge detector or using features to train an edge detector, both yield unsatisfactory results when applied to a wide range of environments or scenes. The best solution to this problem would be to find the universally correct definition of an edge and from there a universally correct edge detector could be developed. However, this does not seem possible given the current level of understanding and knowledge in the field. The approach which this thesis will take is an extension of the training approach. Rather than creating a single filter, or set of filters, which will solve a particular type of image, a framework for creating or matching specialized filters to particular scenes will be created.

In this thesis a framework is developed for creating a system which will match an input image with its corresponding edge filter. A system based on this framework will perform better than current standard methods across a wide range of images from different environments or scenes.

The remainder of this document is organized as follows: Chapter 2 describes the background concepts necessary to understand and carry out the system outlined in this work. Chapter 3 describes the overview of the edge filter generation framework and then goes

into detail explaining each individual component. The explanation of each component is in two parts: first the purpose of the component and what type of methodology can be used to fulfill it is described, and then the particular approach to be used for the prototype systems is given. The results of the prototype Edge Filter Creation Systems are then given and compared to other edge detection methods in Chapter 4. Conclusions are drawn in Chapter 5. Future research to be done on this framework and systems are discussed in Chapter 6.

Chapter 2

Background

This section is intended to give the background information needed to understand this research's proposed method and subsequent results.

2.1 Image Processing

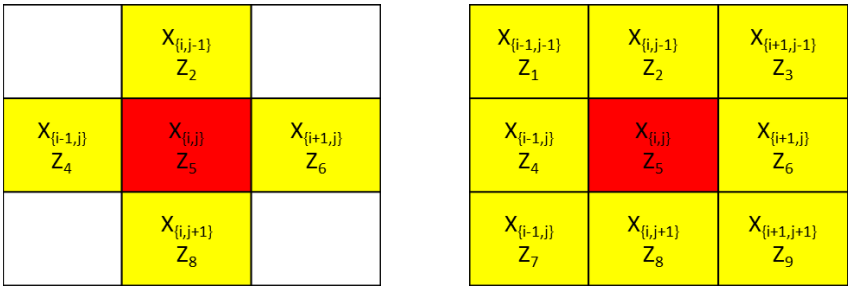
Image processing is a discipline of signal processing. It is considered to be any methodology which takes as its input an image and returns a modified version of that image or a set of features from the image or both. This research is focused on the creation of an edge detection system. This is a fundamental image processing concept were the edges in an image are determined using features from the image itself and are output as a monochromatic image containing only the edges. The remainder of this section will outline image processing concepts related to this research.

2.1.1 Neighborhoods

In the context of image processing a neighborhood is the collection of pixels which is considered when evaluating the next state of a pixel of interest. The two most common neighborhoods, Von Neuman and Moore, are given by Figure 2.1. A Von Neumann neighborhood has the pixel of interest at its center and the neighbors are considered to be every pixel that shares an edge with the pixel of interest. A Moore neighborhood contains

the pixel of interest at the center and the neighbors are considered to be every pixel that shares an edge or a corner with the pixel of interest. Many other neighborhoods exist, each used for different purposes under different circumstances. This research is concerned with neighborhoods containing only in a 3x3 pixel neighborhood.

Figure 2.1 Von Neumann and Moore Neighboorhoods



(a) Von Neuman Neighborhood (b) Moore Neighborhood

X gives the absolute position, Z gives the neighborhood position, relative to the pixel of interest

When dealing with neighborhoods it is important to consider what occurs when the pixel of interest is at an edge of the image. These pixels are called *extreme pixels*. For an extreme pixel, a number of the pixels in its' neighborhood extend off of the image which render any operation depending on those pixels invalid. There are two primary approaches for dealing with this occurrence. The first is known as a null boundary (NB) solution. This solution simply assigns each neighborhood value which has extended off of the Image to be 0. This is equivalent to zero padding the entire image so as to make every extreme pixel have a valid neighborhood. The other solution is called a periodic boundary (PB). This type of boundary assumes that the extreme pixels on each side are adjacent to each other. For example: if the extreme pixel of interest was (3,0) then the pixel (3,-1) would be pixel (3, *last column*) [12]. In this research a periodic boundary is used for neighborhood operations.

2.1.2 *Edge Image* Characterization

A hard problem in edge detection is determining how good an *edge image* is. This is a necessary evaluation, without it it would be impossible to determine if one edge detection method was superior to another. Being able to rate edge detectors is particularly important when training them because a fitness function is needed.

The place to start on this problem deciding what constitutes a good *edge image*. Canny gave one of the early definitions [13].

- **Good detection**: There should be a low probability of false negatives and false positives. This maximizes the signal to noise ratio.
- **Good Localization**: The marked edge should be as close as possible to the center of the true edge to be detected.
- **Single response**: There should be only one edge pixel detected for every true edge pixel that exists.

From this definition a number of methods for evaluating *edge images* have been developed. The current methods for *edge image* characterization can beneficially separated into three types of methods: non-reference based measures, human evaluation, and reference based measures. Non-reference based measures typically suffer from many biases and do not use much information from the original image and as such will not be considered. Human evaluation, though more reliable than non-reference based measures, is impractical because it is often challenging to use for an automation application [14]. In order to allow for the development of this automation framework, a reference based measure will be used

for development and evaluation with human evaluation used only to supplement in the final analysis.

Reference based methods use a gold standard or ground truth (GT) to compare generated *edge images* against. At their core these methods are concerned with only two types of fundamental errors. Type I errors are false positive edge pixels (FP), pixels which are marked as edges by the detector but are not edge pixels according to the ground truth. Type II errors are false negative edge pixels (FN), pixels which are marked as edge pixels in the ground truth but not by the edge detector [15]. The remainder of this section will give an overview of a few of the methods that have been used.

Hausdorff Metric

The Hausdorff metric is given by equation 2.1 and essentially gives the maximum distance between an estimated pixel and its corresponding edge. The smaller the number the better the images match. Though neither this author nor Baddeley have been able to find reference of its use for *edge image* comparison it is interesting for a theoretical basis [15].

$$H(A, B) = \max sup_{x \in A} d(x, B), sup_{x \in B} d(x, A) \tag{2.1}$$

- A is the Ground Truth Image with edge pixels marked as 1
- B is the calculated *edge image* with edge pixels marked as 1
- $d(\alpha, \beta)$ is the distance from α to the closest pixel in β

Misclassification Error Rate

Misclassification error rate is the simplest of the error metrics. It is the summation of all of the type I and type II errors in the *edge image* normalized relative to the total number of pixels [15] as given by equation 2.2. Simply stated it is the fraction of misclassified pixels. This metric also indicates better performance with a lower score.

$$\epsilon(A, B) = \frac{n(A \Delta B)}{n(X)} \tag{2.2}$$

- A is the ground truth *edge image*
- B is the calculated *edge image*
- X is the image itself
- n represents number of pixels in the set

Pratts Figure of Merit

Pratts Figure of Merit (FOM) is the most commonly used and accepted standard for *edge image* evaluation. It is given be equation 2.3. This equation attempts to assign an intelligent error rating to the number of type I and II errors as well as address the localization problem [16]. This metric gives a rating between 0 and 1, 1 being the best.

$$R = \frac{1}{I_N} \Sigma_{i=1}^{I_A} \frac{1}{1 + a * d_1^2} \tag{2.3}$$

- $I_N = MAXI_I, I_A$
  - I_I is the number of ideal *edge image* pixels

  - I_A is the number of actual *edge image* pixels

- a is a scaling constant which is adjusted to penalize lack of localization

- d_1 is the distance between a predicted edge point and the nearest ground truth edge point measured along a line normal to the nearest line of ground truth edge points.

From this single method there have been a number of improvements all of which report results in the same way. One well known weakness of the FOM is that it does not adequately account for type II errors. In particular, the location of the error pixel with respect to the true edge is not taken into account. This causes unfair error in otherwise successful images. Pinho et al. gives an extension to compensate for this discrepancy in equation 2.4, note the definition of d changes [17].

$$R = \frac{1}{N_T}\Sigma_{i=1}^{N_T}\frac{1}{1+a*d_2^2}\frac{1}{1+\frac{\beta*N_{FM}}{N_T}} \tag{2.4}$$

- N_T is the number of ground truth edge pixels

- a is a scaling constant which is adjusted to penalize lack of localization

- d_2 is the distance between a ground truth edge point and the nearest predicted edge point.

- N_{FM} is the number of false positive points

- β is the scaling factor for the false positive edges.

A further modification of that method was given by Wenlong et al. on the rationale that the recall of edge information is accounted for by d_1 and the precision is accounted for by

d_2, see equation 2.5 [16].

$$R = \frac{1}{N_{T\cup P}} \Sigma_{i=1}^{N_{T\cup P}} \frac{1}{(1 + a * d^2)(1 + a * d_2^2)} \tag{2.5}$$

- $N_{T\cup P}$ is the number of predicted of truth edge pixels.

Baddeley Delta Metric

The Baddeley Delta Metric (BDM) is a separate approach to correct a number of weaknesses in the FOM as well as other method of *edge image* comparison [15]. The comparison is accomplished in this method by measuring the similarity of subsets of featured points, typically represented by the value 1. The measure adaptation for binary *edge images* is taken from a paper written by Uguz et al. [18]. For this measure the lower the score the better the *edge image*.

Let B_1 and B_2 be two binary images with the same dimensions $M \times N$, and let $\Omega = \{1, ..., M\} \times \{1, ..., N\}$ be the set of their positions. Given a value $1 < k < \infty$ the k-BDM between the images B_1 and B_2 (denoted $\Pi^k(B_1, B_2)$) is defined as: [18]

$$\Pi^k(B_1, B_2) = [\frac{1}{|\Omega|} \sum_{t\in\Omega} |w(d(t, B_1) - w(d(t, B_2)|]^{\frac{1}{k}} \tag{2.6}$$

- $d(t, Bi)$ is the Euclidean distance from the position t to the closest true edge featured point of the image.
- B_i and $w : [0, \infty] \rightarrow [0, \infty]$: A concave, increasing weighting function
- $w(x) = x$
- $k = 2$

These are but a few of the related methods for edge detection classification, many others exist [9] [19] [20] [14].

This research uses the BDM. This metric was developed in order to improve upon other existing methods, specifically Pratts Figure of Merit, misclassification error rate, and the Hausdorff metric. Baddeley gives very thorough reasoning as to how each of these metrics are limited and how the BDM overcomes their limitations. The largest limitation to consider is that of the FOM. The limitation of the FOM is its inability to handle localization errors in any manner other than adjusting a scaling factor [15]. Essentially, the FOM does not have the capacity to address the pattern of the error but only the average displacement or the edge pixels from the anticipated ground truth positions. The Hausdorff metric cannot be used because it only gives the maximum error pixel and hence only needs one outlier to skew the image results. The misclassification error rate is not useful for an edge detection application because it does not concern itself with how close the error pixels are to correct, only that they are errors.

2.1.3 Standard Edge Detection Methods

Edge detection is a fundamental problem in image processing, one which many other techniques rely upon in order to function properly. As such, a great deal of research has been done on edge detection are over the last few decades. Of the many approaches that have been developed, there are a few which have emerged as the 'standard approaches' for the edge detection problem. These approaches work well in most instances: Sobal, Canny, Roberts, Prewitt, Laplacian of Gaussian (LoG). All of these methods are based on the

differential gradient of the image intensity. The following sections summarize these techniques.

Sobel

This filter is named for Irwin Sobel. It is comprised of two orthogonal filters given by Figure 2.2. These filters are convolved with the original image resulting in an approximation of the image intensity gradient for two orientations. These orientations are typically 0 and 90 degrees. The two resulting images from the convolution are overlaid to give a final *edge image*. This process is summarized in equation 2.7 [10].

$$\begin{aligned} \nabla f &= \sqrt{g_x^2 + g_y^2} \\ \nabla f &= \sqrt{[(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)]^2 + [(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)]^2} \end{aligned} \tag{2.7}$$

- g: Neighborhood to be convoluted against (kernel)
- z: Neighborhood location
- ∇f: Resulting intensity gradient image

If a monochromatic image is required a threshold T is then used on the final *edge image* to separate edge pixels from non-edge pixels as in equation 2.8.

$$\nabla f \geq T \tag{2.8}$$

Figure 2.2 Horizontal and Vertical Sobel Kernels

-1 z_1	0 z_2	1 z_3
-2 z_4	0 z_5	2 z_6
-1 z_7	0 z_8	1 z_9

(a) Horizontal

-1 z_1	-2 z_2	-1 z_3
0 z_4	0 z_5	0 z_6
1 z_7	2 z_8	1 z_9

(b) Vertical

Canny

The Canny filter is a methodology which builds on the Sobel filter by incorporating some pre and post processing steps to increase *edge image* detection accuracy. The Canny edge detection approach is accomplished in 5 general steps [10].

- **Image Smoothing**: Typically accomplished using a two dimensional Gaussian filter as in equation 2.1.3.
- **Intensity Gradient**: The Intensity Gradient is determined by convolving a Kernel similar to the Sobel filter in Figure 2.2.
- **Suppression Weak Edge Elimination**: Because of the nature of the gradient detection, it is often the case that a given edge will be found to have a width of greater than one pixel which is not ideal as mentioned in Section 2.1.2. The local maximum gradient value in a given direction in a group of edge pixels is assigning to be the edge pixel. This allows for edges of single pixel width to be created.
- **Double Threshold**: The remaining edge pixels are categorized as weak and strong

edges based on two intensity threshold.

- **Track Edges**: Weak edge pixels are included using heuristics to complete strong edges.

$$G(x, y) = e^{-\frac{x^2 + y^2}{2\delta^2}} \tag{2.9}$$

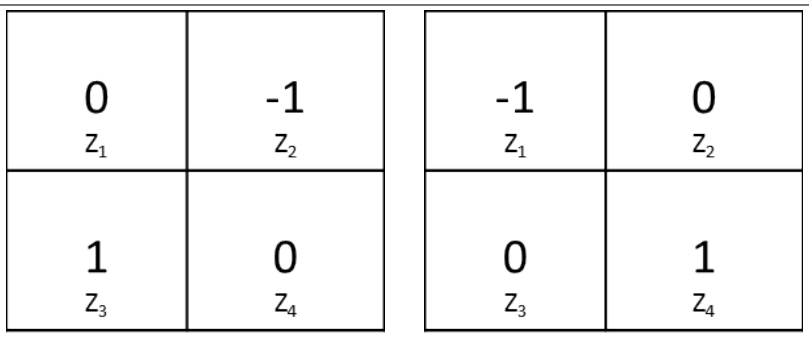
Roberts

The Roberts filter, also known as the Roberts cross, accomplishes edge detection by taking a discrete differentiation of a localized neighborhood to find intensity variations. This differentiation is done by convolving the two $2x2$ filters given by Figure 2.1.3 individually against an input image and then merging the two results as given in equation 2.10 [10]. This method is limited in that it cannot detect edges that are multiples of 45° but it is extremely fast to implement in hardware.

$$\begin{aligned} \nabla f &= \sqrt{G_L^2 + G_R^2} \\ \nabla f &= \sqrt{[z_4 - z_1]^2 + [z_3 - z_2]^2} \end{aligned} \tag{2.10}$$

- g: Neighborhood to be convoluted against (kernel)
- z: Neighborhood location
- ∇f: Resulting intensity gradient image

Figure 2.3 Roberts Cross Filters



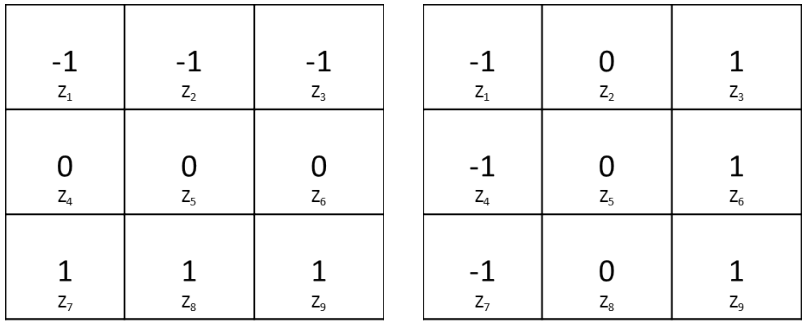
(a) 45° filter (b) 135° filter

Prewitt

The Prewitt operator is a combination of the Roberts and Sobel approaches. It uses two orthogonal 3X3 neighborhoods like the Sobel methodology. However, Prewitt combines them in the same manner as Roberts cross by convolving the two filters individually against an input image and then merging the two results as given in equation 2.11 [10]. The filters can be seen in Figure 2.4.

$$\begin{aligned} \nabla f &= \sqrt{g_x^2 + g_y^2} \\ \nabla f &= \sqrt{[(z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)]^2 + [(z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)]^2} \end{aligned} \tag{2.11}$$

Figure 2.4 Prewitt Filters



(a) Vertical (b) Horizontal

Laplacian of Gaussian

The Laplacian of Gaussian (LoG) filter applies a $\nabla^2 G(x, y)$(equations 2.12 and 2.13 kernel, as given by equation (2.12), to an image. This filter has two effects, one it smooths the image to reduce the noise using the Gaussian in the kernel and simultaneously applying the Laplacian for differential edge detection. This yields a double *edge image* in which the zero crossing between the edges is found to gain the final *edge image* [10]. A typical Laplacian of Gaussian filter is given by Figure 2.5.

$$L(x, y) = \frac{d^2 f(x, y)}{dx^2} + \frac{d^2 f(x, y)}{dy^2} \tag{2.12}$$

$$\begin{aligned} \nabla^2 G(x, y) &= \frac{\partial^2 G(x, x)}{\partial x^2} + \frac{\partial^2 G(y, y)}{\partial y^2} \\ \nabla^2 G(x, y) &= [\frac{x^2 + y^2 - 2\delta^2}{\delta^4}] e^{-\frac{x^2 + y^2}{2\delta^2}} \end{aligned} \tag{2.13}$$

Figure 2.5 Laplacian of Gaussian Filter

0 z_1	1 z_2	0 z_3
1 z_4	-4 z_5	1 z_6
0 z_7	1 z_8	0 z_9

2.1.4 Color Vector Field Edge Detection

Another way to define an image is a function which maps a spatial field into a color field. Using this definition it is possible to generalize the gradient of a scalar field to the derivatives of a vector field in order to ultimately obtain a representation of the largest changes over a distance in the spatial field. The theory in full is given by Lee and Cok in [48] and is summarized below.

The gradient of a vector is determined by taking the partial derivatives of the quantity of interest with respect to each dimensional direction. In general this is given by equation 2.14.

$$f'(x) = D(x) = \begin{bmatrix} D_1 f_1(x) & \cdots & D_n f_1(x) \\ \vdots & \ddots & \vdots \\ D_1 f_m(x) & \cdots & D_n f_m(x) \end{bmatrix} \tag{2.14}$$

Where f(x) is the function mapping into the desired space with dimensions $1 - m$, and D is the partial derivative with respect to the dimensions $1 - n$. Vector theory states that when traveling from a given point with a unit vector u in the spatial domain then $d = \sqrt{u^T D^T D u}$ will be the corresponding distance traveled in the transformed domain described by d. The largest Eigen vector Eigen value pair from $D^T D$ will result in a maximum of d. This implies that the largest Eigen vector Eigen value pair are the vector fields gradient direction and gradient magnitude respectively. This is the vector gradient. The vector gradient is best found using singular value decomposition (SVD). The SVD is used to map a matrix of values into a new space which has cardinal direction which incorporate

a maximum of unique information along their axis. This is reflected by the relative sizes of the Eigen values. The largest Eigen value will thus contain the highest concentration of unique information; in this case the information is edge information. For a two dimensional image in the Y, C_r, C_b space the matrix of partial derivatives is given by equation 2.15

$$D = \begin{matrix} \frac{\delta_y}{\delta_x} & \frac{\delta_y}{\delta_y} \\ \frac{\delta_{C_r}}{\delta_x} & \frac{\delta_{C_r}}{\delta_y} \\ \frac{\delta_{C_b}}{\delta_x} & \frac{\delta_{C_b}}{\delta_y} \end{matrix} \tag{2.15}$$

In order to simplify the following variables are defined in equations 2.16-2.18.

$$p = (\frac{\delta_y}{\delta_x})^2 + (\frac{\delta_{C_r}}{\delta_x})^2 + (\frac{\delta_{C_b}}{\delta_x})^2 \tag{2.16}$$

$$t = (\frac{\delta_y}{\delta_x})(\frac{\delta_y}{\delta_y}) + (\frac{\delta_{C_r}}{\delta_x})(\frac{\delta_{C_r}}{\delta_y}) + (\frac{\delta_{C_b}}{\delta_x})(\frac{\delta_{C_b}}{\delta_y}) \tag{2.17}$$

$$q = (\frac{\delta_y}{\delta_y})^2 + (\frac{\delta_{C_r}}{\delta_y})^2 + (\frac{\delta_{C_b}}{\delta_y})^2 \tag{2.18}$$

This makes the $D^T D$ matrix take the form given by equation 2.19

$$D^T D = \begin{bmatrix} p & t \\ t & q \end{bmatrix} \tag{2.19}$$

The largest Eigen value which results from the use of SVD is given by equation 2.20.

$$\lambda = \frac{1}{2}(p + q + \sqrt{(p+q)^2 - 4(pq - t^2)}) \tag{2.20}$$

The square root of this value gives the magnitude of the gradient vector at every point of the image, which is used as the edge detection criteria. This λ value is assigned a threshold to separate the edge pixels from the none edge pixels.

2.1.5 Learning Approaches to Edge Detection

In recent years, research has branched out from solely gradient and differential based approaches to edge detection and has begun to focus on computer learning approaches to training edge detectors.

There are a large number of methods proposed which use a consensus of a large number of edge detection techniques to achieve an improved edge detection [21] [11] [22] [23] [24]. The most interesting research imposed global constraints as well as additional weighting logic to improve results by rejecting false edges [25]. The thought in all these approaches is that by polling many edge detectors the specificity of each will be averaged out to create a versatile edge detector. The unifying theme of all of these papers is their exhaustive approach to edge detection. By using several processing techniques these methods will take longer to arrive at an *edge image* which is undesirable.

One approach that holds potential is training edge detection systems using Boosting. Kokkinos et al. uses F-measure, which is the ratio of the product of the precision and recall of a function to its sum, and a boosting variant called ANY-Boost to create and train sets of weak learners to solve the edge detection problem [19]. This method of boosting is an improvement on another earlier boosting method, Filterboost [26]. The most important improvement is that the enhanced method handles ambiguity in orientation by using a multiple instance learning technique and leveraging a larger training set then the original

method. This design yields a set of filters which will be applied together and give an improved edge response over current methods for the particular trained scene class. The use of multiple filters to obtain the *edge image* increases the computational time making this method less desirable.

Lopez-Molina et al. uses fuzzy logic to augment the parameters of the Canny edge detection method [27]. The reported results show that this method gives a large improvement over the standard Canny method. As part of the research the need for an unsupervised learning approach for parameter assignment is identified. The paper proposes a histogram based technique to assign filter parameters to filters based on image characteristics. Three implementations are given. Lopez-Molina et al. gives strong indication that fuzzy logic approaches will work well for enhancing edge detection methodologies.

Another type of approach taken is to apply statistics to edge detection. Koern and Yitzhaky use a saliency map which is incorporated with χ_2 analysis of statistical measures of *edge images* from multiple filters to select the best parameters for the Canny edge detector [22]. Konishi et al. used traditional edge detectors on the Sowerby and South Florida data set to train a multidimensional probability histogram to be used in two applications [9]. The first application was to use the distributions in conjunction with a trained set of traditional filters on color images at various image scales to choose improved parameters for the traditional filters; Canny was used as the experimental filter. The second was to use the distributions in conjunction with Garbor orientation filters and log transformations, again on Canny for experimentation. All the methods resulted in an improvement over the Canny edge detector based on their metric for success. The statistical measures are another

training strategy for assigning an image to its edge filter based on characteristics of the image itself.

Wang et al. proposes a neural network approach based on a concept of spatial moments [28]. The network is trained based on a 5x5 neighborhood as the input. The output is whether or not the central pixel is an edge pixel. This method claims greater efficiency than the LoG method while giving increased results. The results are given against *Lena* but with no empirical evidence to support improvement. When investigated the results from the method and LoG are very similar. This method does raise the point that a neural network is fast computationally but a more rigorous application of comparison criterion is needed. Wang et al. does indicate that a neural network when properly trained could solve *edge images* robustly for one scene class, and potentially more if the proper topology and training set is identified.

Wenlong et al. uses a genetic algorithm approach to train an edge detector for a single image [20]. That edge detector is then used on other images within the same scene class to good effect.

Komati et al. gives an interesting approach which is inspired by the two channeled way the human mind perceives edges [29]. This approach uses a standard edge detector coupled with a region growing technique. These two methods check each other to accurately find the edges of an image. This method gives good results but includes further processing after a filter is applied making it less desirable. However, this methodology could also be used in this research as an edge detection filter generator in the framework.

Selection of Edge Detectors (SED), a system devised in 1998, is another interesting

concept. SED contains a bank of filters with descriptions of their capabilities which will be given as output to the system. This system takes as input an edge, an image where the edge is taken from, and constraints on the quality of its detection in the image. The system then analyses its filter bank and matches the input edge with the filter which will best find it in the provided image [30]. This system methodology is similar to that of the prototype approach.

2.1.6 Cellular Automata Applications

The methods of edge detection already mentioned in this section have the ability to be trained or modified to suit different environments. These methods all offer a degree of versatility which could be used in the proposed system. However, it is hypothesized that the larger the filter space is the better the system will be able to train for a variety of scenes. Section 2.2 outlined that CA have a very large filter space and that this space contains many automamta capable of complex computations. It has further been shown in many papers that CA have many applications in image processing particularly for edge detection [31] [32] [12] [33] [34] [18].

Thomas, in his thesis, investigated the potential of CA in a variety of image processing applications [31]. On the topic of edge detection, a genetic algorithm training approach was taken for specific and generic images using non-uniform CA. The results for specific images were promising. For generic images the results were lacking. This is due to the poor adaptation of the genetic algorithm, the small sample size used to train, and the expectation that a single filter could solve multiple image types. This work will show that generic image edge filtering is possible when approached in a different manner.

Fasel investigated a limited set of CA which are referred to in the paper as Linear CA [32]. These CA are two dimensional binary CA which are achieved using the EX-OR operation only. The paper investigates these 512 transition rules and finds that many of them have strong edge detection capabilities. Uguz et al. took this research to its next logical step and tested the CA rules which had strong edge detection and compared the results with standard edge filtering methods [12]. The paper found that for the well known test images such as *Peppers* and *Lena* the CA filter was able to out perform the standard methods. Mirzaei et al. employs a fuzzy rule set with CA in order to obtain *edge images*. These Edge detectors also have the property of noise resistance [33]. For that system, the fuzzy CA filters yielded superior results when compared in a subjective way with Sobel and Roberts filters. Sinaie et al. also uses fuzzy set theory with their own learning automata algorithm to train edge detectors based on CA [34]. A later paper published by Uguz et al. uses fuzzy set theory and PSO to train filters which work better than conventional methods on particular images [18]. In short, the rule sets of CA have been shown many times to be capable of solving the edge detection problem when trained.

This research will leverage the approaches used by Uguz et al., Non-Fuzzy CA and Fuzzy, as the filter generators for the proof of concept of the prototype filter generator system.

2.1.7 Principle Component Analysis

Principle component analysis (PCA) is a method of dimension reduction. It is common for features extracted from a data set to be highly coupled. The interdependence of the data in

the data set creates redundancies in computation, which will greatly increase overall computation time in future processing steps. It is often beneficial to remove these redundancies in order to to improve performance in those future steps. PCA takes a set of features and maps them to a lower dimensional space [35]. The co-variance method which is used for this research is done using the following steps.

- First the mean, μ, of the data is calculated and then subtracted from the data set. By centering the mean the accuracy of the mean squared error, which is used for component selection, is increased.

- The covariance matrix, C, of the input data, X is found by taking the outer product of the matrix with itself as given by equation 2.21.

$$C = E[(X - \mu)(X - \mu)^T] \tag{2.21}$$

- The eigenvalues and eigenvectors of this co-variance matrix are determined using singular value decomposition (SVD), which is defined by Theorem 1. This is possible because eigenvalues are singular values for the covariance matrix because it is positive semidefinite.

- The Eigen pairs are arranged by magnitude, where the magnitude of the eigenvalue reflects the energy, or percentage of information, contained in the corresponding dimension.

- The first n eigen values are taken as the new dimensions where the sum of the energy in the first n eigen values yields the desired energy to be maintained in the reduced

dimensions. Typically the desired energy is greater than 90%.

- A matrix W is constructed using the number of eigen vectors from the first n vectors in V which will achieve the energy goal specified. this W is now the transformation matrix from the original data set to the new reduced feature space.

- $Reduced \quad Feature \quad Matrix = A * W$

Theorem 1 *Let A be a mxn real or complex matrix with rank r. Then there exist unitary matrices U(mxm) and V(nxn) such that:* $A = U\Sigma V^H$ *where* Σ *is an mxn matrix with entries:*

$$\Sigma_{ij} = \sigma_i if i = j, 0 if i \neq j \tag{2.22}$$

the quantities $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r \geq \sigma_{r+1} = \sigma_{r+2} = ... = \sigma_n = 0$. *are called the singular values of A [36].*

2.1.8 Scene Recognition

Scene recognition is a field of research in image processing which has been receiving an increasing amount of attention. This research has been heavily motivated by the fields of video processing and robotics. Applications range from localization and adaptive interaction for robotics, to indexing for video storage. Scene recognition is useful for essentially any application that can be enhanced by having foreknowledge of the location, or type of location.

Scene recognition falls into two processing categories: recognition and classification. The first asks the question have I seen this location before? Essentially the algorithm wishes to know if the scene it is currently looking at is a place that it has seen before. The second type is scene categorization. Instead of asking if the system has seen this particular location before the algorithm asks if it has seen a scene similar to the one currently being presented.

For example the first algorithm will identify the house in which you live and tries to return positive for only that house, the other will return positive for all houses.

The challenges presented by either of these scene recognition questions are substantial. Scene recognition must function across various scales, from various perspectives, and with varying lighting. In addition, scene recognition must also deal with the inherent ambiguity and variability of what constitutes a particular scene.

Every scene recognition approach has two essential steps, the feature extraction and the classification. Feature extraction is the process through which relevant information is extracted from an image which will aid in the classification of an image. The objective is to obtain features which are scale, orientation, and illumination invariant. These features are then used to train a classification method. Many schools of thought have been applied in an attempt to discover an effective feature se for scene recognition. An extensive list of these approaches in summary is given by Jianxiong et al. [37]. A few methods will be explicitly discussed to illustrate the reasoning for choosing the specified approach.

Quattoni and Torralba uses a region based approach. Rather than searching for particular objects in a scene the researchers uses two types of features [38]. The first is a holistic GIST descriptor which describes an image as a whole. The second is a collection of scale invariant feature transforms (SIFT) grouped by region and arranged as visual words. It is thought that similar scenes will have similar regions which are in spatially similar locations. Thus it is possible to select regions that are important for a particular scene. The training of the system uses a support vector machine (SVM). The reported results are an accuracy of at best 63% on indoor systems and up to 95% for outdoor scenes, depending on the classes.

This system still relies on the researcher to select regions which are important by hand in the training images. This approach did yield an improvement for the papers proposed problem, indoor scene classification, however having to select a region of interest by hand is not ideal for a automating a process.

The approaches to scene recognition that will best suit the application of this thesis are those that use automated algorithms to extract low level features. Lazebnik et al. uses a combination of SIFT and GIST features to find a scene categorization [39]. The image is separated into 4x4 regions which act like the regions selected by Quattoni and Torralba, except that they can be generically applied. Both methods use (SVM) for classification. The results for recognition between 15 classes for categorization was between 60% and 90% accuracy. It was shown that for indoor features that the percentage was always significantly lower then outdoor features. This is hypothesized to be because of the variability of indoor scenes within the same class. The same features were used by Madokoro et al. However, an adaptive resonance theory network was used for training so that the scene recognition could be unsupervised [40]. The results for robotic localization on five classes in this system where between 65% and 95%. It is interesting to note the similarity in results between Lazebnik et al. and Madokoro et al. They both use the same features but their task are very different. Lazebnik et al. attempted to categorize the scene, while Madokoro et al. was trying to recognize a specific location that has previously been visited. Despite this their results are very similar, even with different numbers of classes being investigated.

Even simpler approaches are used which incorporate only a single holistic feature extraction approach. Pavlopoulou and Yu compare the effectiveness of edge gist descriptors

to that of gist descriptors which use the entire image [41]. It was found that edge gist out performed image gist for indoor scenes but not for outdoor scenes. For outdoor the best correct categorization was 80% and for indoor it was 70%. Xianglin and Zheng-Zhi use a texture based gist approach which leverages the census transform in a novel way to describe an image [42]. This method returns a 71%-95% correct classification for an 8 class system. Siagian and Itti in two papers outline a system of GIST classification which uses color and edge orientation to describe an image [43] [5]. This system investigates the ability to determine different scenes but also distinguishs between different views or segments of the same scene for localization. It reports accuracy between 85% and 90% at this task. These simplest methods are the least computationally intensive and also use the smallest number of assumptions making them ideal for a pre-processing step. It is for that reason that the processes outlined by Siagian and Itti will be used to as the feature extraction method for this thesis.

2.2 Cellular Automata

Cellular Automata (CA) is a branch of mathematics which concerns itself with the ability of simple systems to arrive at complex outcomes. A CA consists of:

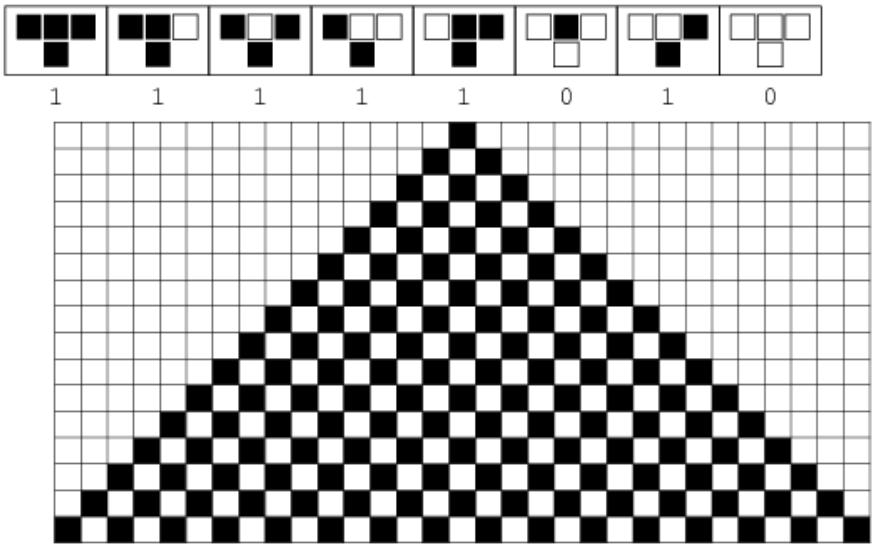
- A set of adjacent cells L, known as a cell lattice
- A finite set of states S which the cells can take
- A neighborhood of cells N which defines a relationship between the cells
- A transition rule or function F which defines how the cells update in discrete time.

These four properties are often represented in a 4-tupleL,S,N,F [32]. The study of these systems began with one dimensional CA which have two states and a neighborhood of three: left, center and right. The number of automata which exists inside of the touple is determined by the number of unique configurations that the function F, also known as transition rules, can take on. This value is given by K^{K^N} where K is the number of states in S and N is the number of neighbors in the neighborhood. Thus for a system where S takes on binary states and the neighborhood is of size three the total number of states is $2^{2^3} = 256$. Hence this set of CA systems came to be named according to the number of their transition rules, which is what Wolfram did as he began to study these systems [44].

CA can also be beneficially examined as belonging to one of 4 different classes. These different classes and the range of behavior which they represent are what give CA their computational power.

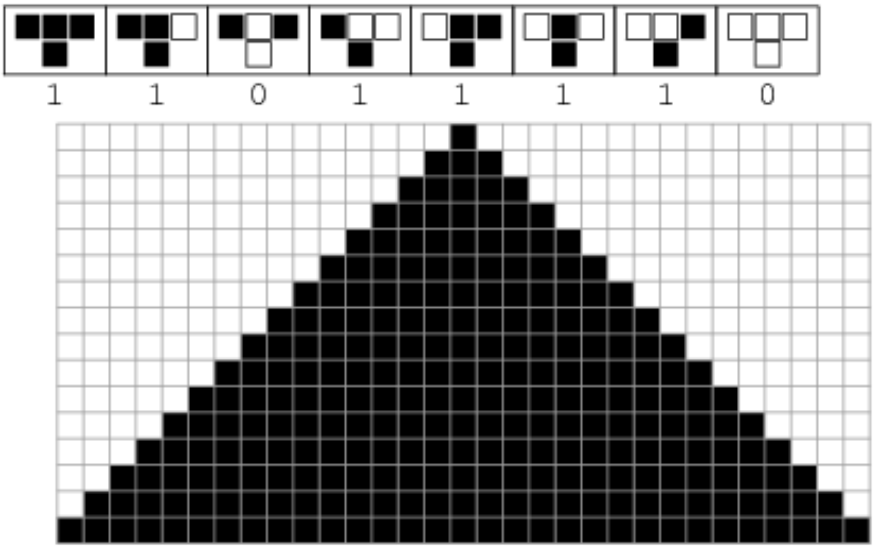
- Class 1: Rules which converge to homogeneity (Figure 2.6).

Figure 2.6 Rule 250 [1]



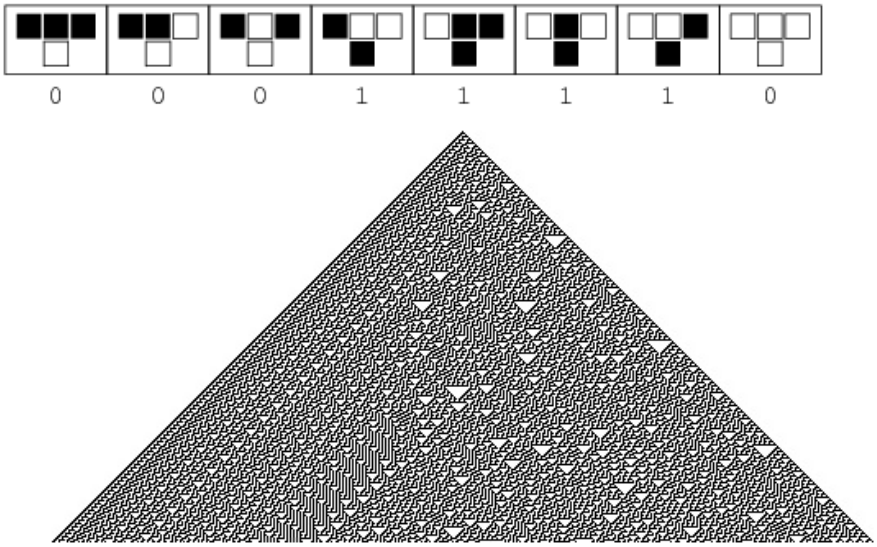
- Class 2: rules which create stable or periodic patterns (Figure 2.7).

Figure 2.7 Rule 222 [2]



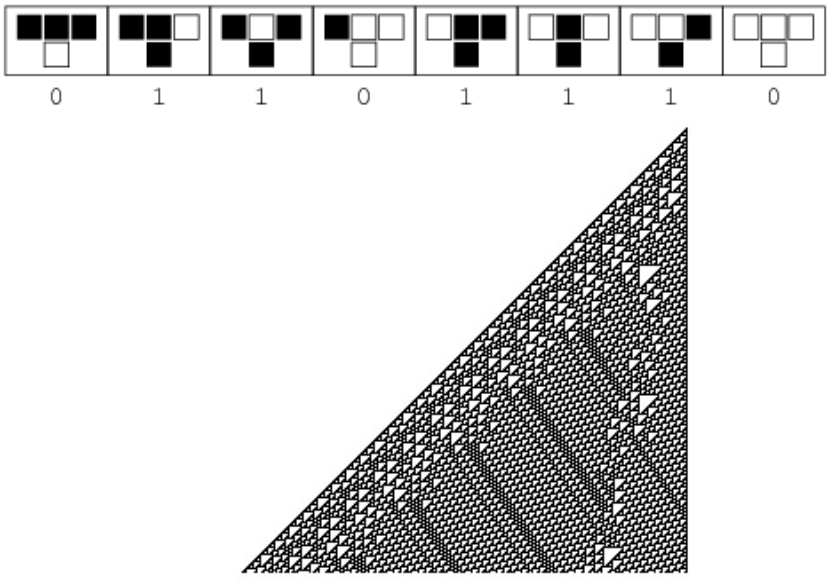
- Class 3: rules which devolve into chaos (Figure 2.8).

Figure 2.8 Rule 30 [3]



- Class 4: rules which generate complex patterns (Figure 2.9).

Figure 2.9 Rule 110 [4]



For this paper two dimensional CA will be applied. These CA extend the rules of one dimensional CA but maintain all of the same basic principles in the following manner.

Rather than the lattice being a single line it is a plane. The neighborhood is a Moore neighborhood. The transition rule set now contains 2^{2^9} rules. The number of states for the cells remains 2 for the purpose of this research. The entire plane still updates in a single time step. What is important is that the 4 classes of Cellular Automata are also represented in two dimensions. This means that in this much larger space there are many complex behaviors to observe and use to solve new problems.

2.3 Learning Systems

A learning system is a program which takes in information and uses it to train itself to perform new or specific tasks. There are many different types of learning systems, this section will give a brief explanation of the systems which will be used in this thesis.

2.3.1 Partical Swarm Optimization

PSO is a learning methodology which is derived from the actions of swarms in nature. Inspiration for these algorithms has been drawn from ant hives, bird flocks, and schools of fish [45]. These algorithms have been used to solve a variety of problems and have been applied with particular success to solve global optimization problems. This methodology is population based and as such has a number of particles moving through an n dimensional space in discrete time steps each with a position and a velocity. The velocity is often influenced by three components:

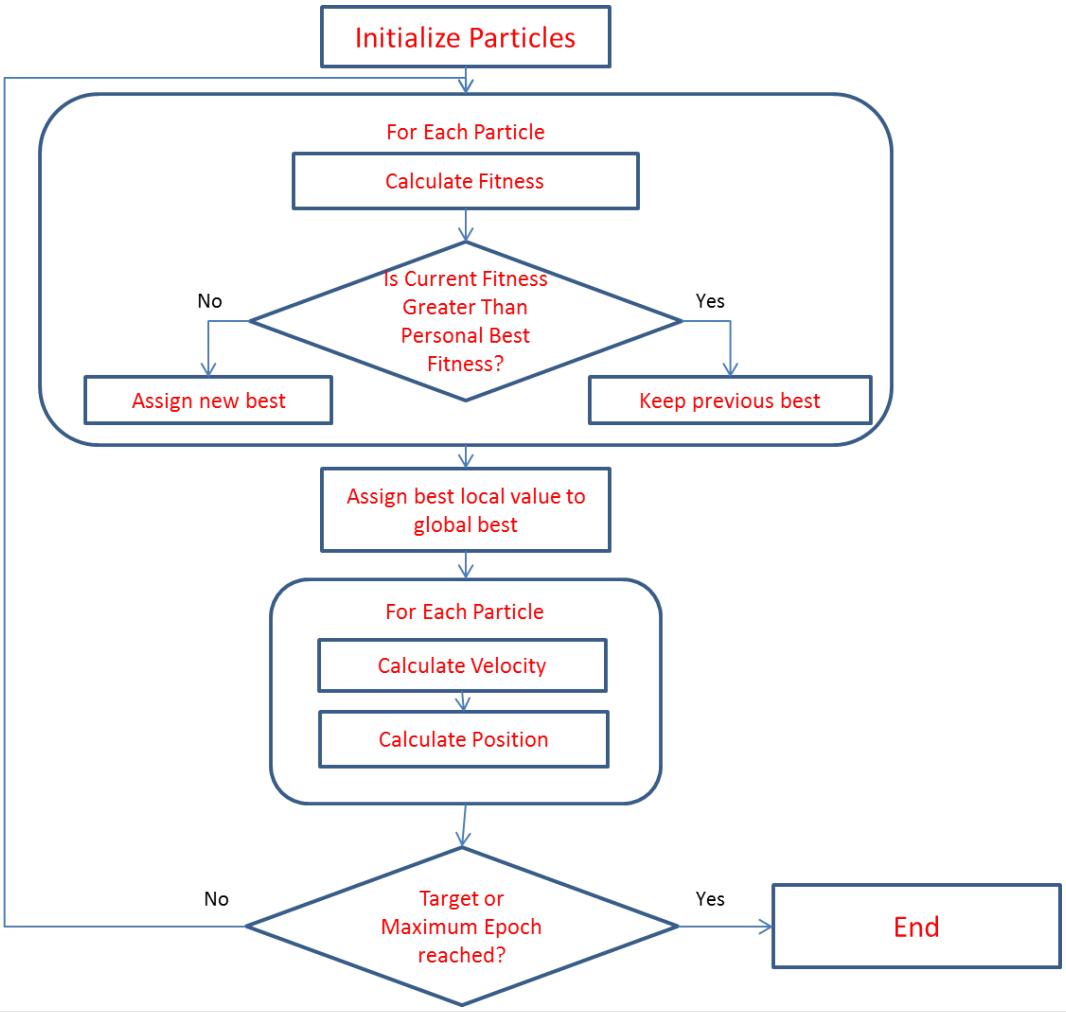
- Current motion influence: The impact that the direction and velocity of the particle has on the direction and motion of the particle in the next time step, virtual inertia.
- Particle memory influence: Each particle knows how well its current location solves

its problem. In addition the particle may have a memory of its previous locations and their fitness. All this information may have an effect on the next position and velocity of the particle.

- Swarm influence: Each particle may also know information about the other particles in the swarm. This information could be global, meaning every particle know the information of every other particle, or it could be neighborhood based, each particle only knows information about near by particles. The information gained in this way also has an impact on the learning of the system.

A typical PSO Algorithm has the form given by Figure 2.10.

Figure 2.10 PSO Algorithm



2.3.2 Neural Network

A neural network (NN) is a learning topology consisting of a directed graph which is patterned after the functionality of the human brain [35]. Typically the network is constructed as a series of layers 2.11. The simplest is a configuration known as a feed-forward single hidden layer neural network. This network has an input layer, a hidden layer and an output

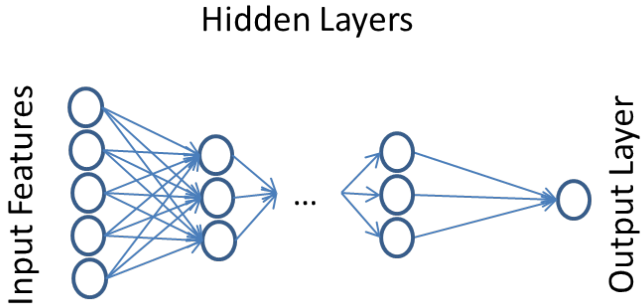
layer. Each layer has a set of neurons (nodes). The nodes in a layer do not connect to each other but each node in a layer will have a connection to every other node in the subsequent layer. Each node also has an activation function which determines if or what the neuron will pass onto the next layer. The input layer of nodes receive the pre-processed features upon which a decision needs to be made. The output layer nodes make the final classification decisions. The hidden layer nodes are present to provide a greater ability to find the separability of the classes. Every neural pathway must lead to an output node. Each connection between a node, is weighted. It is this weighting that allows for the classification to occur. Each layer of the network also has a bias to better tune the classification. Learning occurs when the network is given the capability to reassign the weights of the connections or even change which nodes are connected. The most common learning method is back propagation which is accomplished using the following steps.

- Feed-forward computation: The data is sent through the network.
- Back propagation to the output layer: The output values are compared with the true output values and a delta is determined.
- Back propagation to the hidden layer: the delta is propagated back through the hidden layers of the network
- Weight Updates: using the delta measures the network weights are updated to achieve better performance on a subsequent run.

It is possible to 'over train' a network. When a network is training it is learning the trends in the data which allows for classification. When a system is over trained it has

learned the trends of the training data set but also the noise particular to that data set. An over trained system will not perform well on a new data set because the noise learned in the network will overpower the classification ability of the trends.

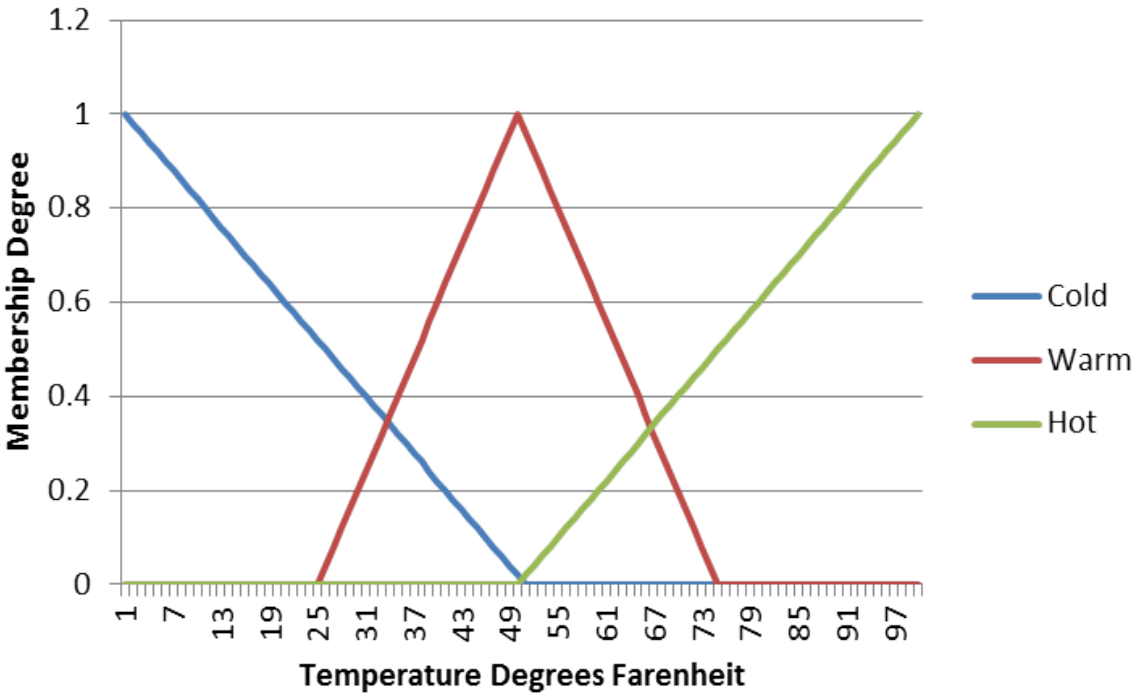
Figure 2.11 Generalized Basic Neural Network Configuration



2.3.3 Fuzzy Transition Rules

This is a type of machine learning which is concerned with modifying the binary rule set to allow for degrees of membership. The degree to which an object belongs to a set is determined by its membership function. This function can be as simple as a step (which is the same as a binary function) but often is a probability distribution which allows for various degrees of ”belongingness” to a set. These membership functions are used to map the value in question to a value in the fuzzy set between 0 and 1. A simple example of a membership function is one for temperature. Instead of the binary hot and cold there can be three classes, hot cold and warm. Their memberships can then be defined as given by Figure 2.12.

Figure 2.12 Temperature Membership Functions



For every value in a fuzzy set there is a fuzzy operator which uses an *IF-THEN* relationship. There is not an else in fuzzy set theory as every value in the set is defined to a specific membership function and every value is mapped in some degree to at least one of the members of the set. The training of a fuzzy logic set is done by training the membership functions so that the proper values are associated with the appropriate fuzzy operator. Lopez-Molina et al. shows that fuzzy logic can be used to enhance edge detection methodologies [27].

2.3.4 Boosting

Boosting is an iterative approach which combines a series of weak learners to generate a strong learner [35]. Typically the input problems are weighted uniformly using some distribution. Each weak learner is trained based on the problem with the highest weight. When each weak learner is trained it receives a weight which is related to its performance. In addition the weight of each problem in the problem set is reassigned to reflect their classification based on the current set of weak learners. The better the classification the lower the weight. Thus the problems with the highest weight are the problems most dissimilar from the ones already solved by the set of weak learner. The training process continues until all of the problems have been solved to a suitable degree of error. The end result is a set of weak learners that collectively solve the problem set.

Chapter 3

Proposed Method

Edge detection algorithms are often employed as feature extraction or pre-processing step in an image processing application [27] [20] [28] [23] [22]. For many applications the environment that is being analyzed is fairly stagnant, and the objects of interest are the only objects in view or the only objects in motion. For situations where the environment is highly controlled, it is not overly challenging to create and tune a filter to give excellent results. There are many papers which already do just that as seen in Section 2.1.5.

There are other situations where the environment that the imaging platform is in is not controlled. One such situation is for a robot which is to be used in, or which will travel to and through, a number of different environments. It is well known that a filter which is tuned to a particular environment will likely perform poorly when entering another. This stems form the fact that the definition of an edge is highly subjective and application specific. Given a set of individuals all looking at the same image it is likely that every individual will arrive at a slightly different interpretation of what the edges of the image are. This problem compounds the issue of edge detection from a computation problem to a learning problem.

For a stationary edge detection task the filter is tuned by the operator for optimal success.

In this case, the learning of the system is done manually by the operator and the machine is hard coded to behave according to the particular definition of an edge in that environment. In order for an imaging platform to be free to move between environments and still have meaningful edge detection, the platform must be able to differentiate between, and know what an edge means in, different environments. This framework is an attempt to address this problem by providing a methodology for teaching a machine what a meaningful edge is for different classes of images. This will allow the machine itself to determine what edge definition to use for a given situation; thus freeing the machine from another facet of low level user intervention.

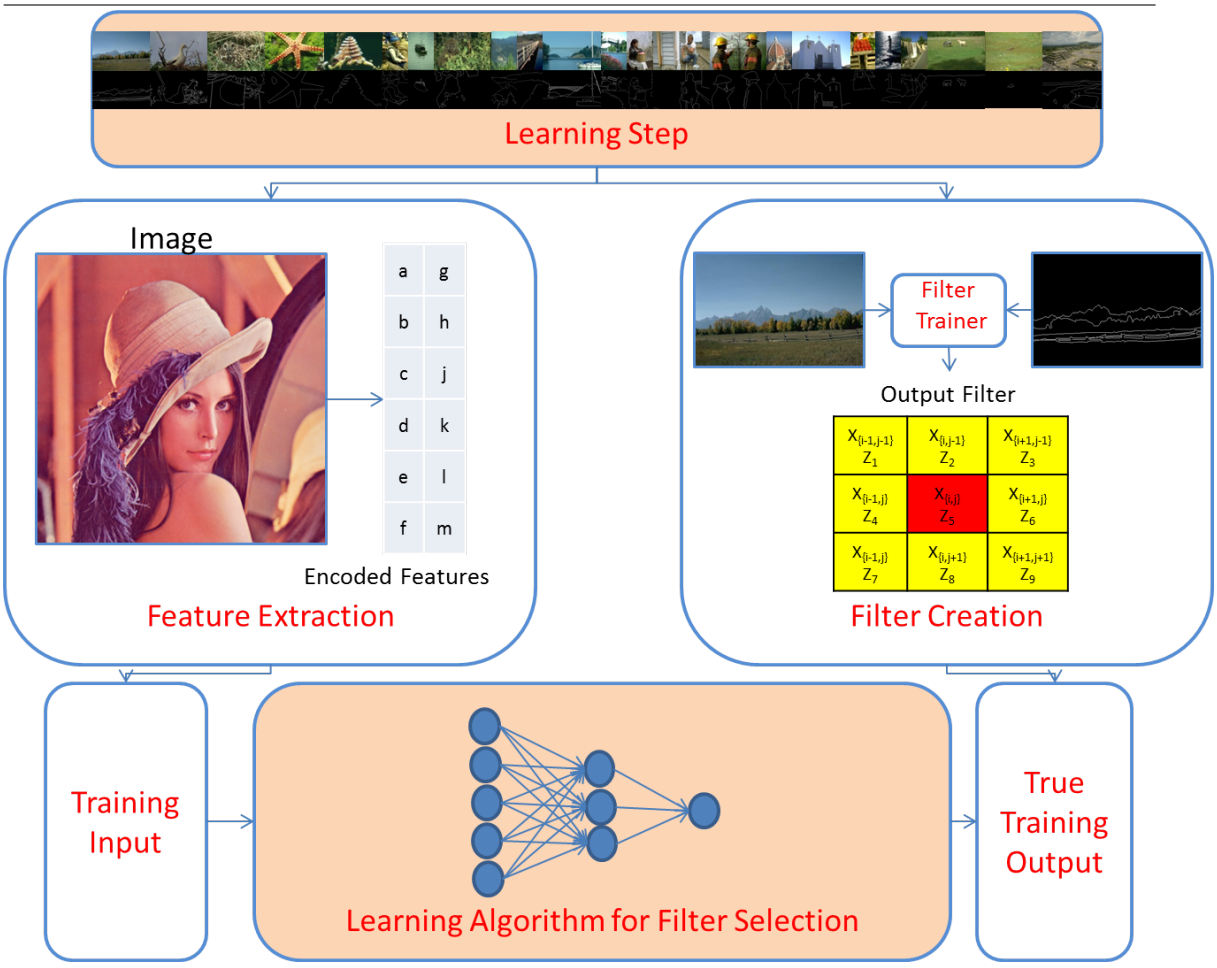
The proposed framework for a system to accomplish this goal has four parts (see figure 3.1), which are modular. The methodology used to fill each part can be replaced, enhanced, or changed in a number of ways to improve the performance of the overall system. These parts are: Learning Step, Image Feature Extraction, Edge Detection Filter Creation, and Filter Selection Method. The end result of the system is a trained system which will associate an input image to a corresponding method for edge detection. This research implements two systems as a proof of concept for this framework. It is left open to future research to create more and improved implementations of this framework.

The success of the proof of concept systems which implement this framework will be measured in the following manner. Each image in a test set will be processed using the standard edge detectors available in MATLAB (Roberts, Canny, Sobel, Prewitt, LoG). Each image will also be processed using the filter selected by the prototype system. The average error of each method will be taken using the BDM index as given in 2.1.2. If the cumulative

error of the implemented system is less than that of all the other methods, the system will be shown to be a viable alternative to the standard methods for diverse image data sets.

The remainder of this section will explain the components that make up the framework as well as the implementation of those components in the two proof of concept systems. Section 3.1 gives what is required of the learning step and the training set to be used for this research. Section 3.2 explains the feature extraction methodology component and the specific method used for the systems is introduced. in Section 3.3 the Filter Generation

Figure 3.1 Proposed Approach Block Diagram



methodologies component is explained with the two implementations used in the proof of concept systems. Finally, in Section 3.4 the requirements for the filter selection algorithm will be addressed and the reasoning behind the method selected for the prototype system will be given.

3.1 Learning Step

The learning step is the method by which the system learns what is and is not considered a good *edge image*. This system will have input image features training to an output of edge filters. The system must have a way of knowing whether the filters it is selecting are correct. Section 2.1.2 gives many different types of methodologies which could be used to fill this role. The methodology used in this research is a reference based measure. Such a measure use a training set with ground truth images and a method for rating the output *edge images*. This decision was made because these types of metrics are the best for automating systems. In addition these types of systems make analysis of results and replication of the system more feasible.

The training set used for this research is the Berkeley Segmentation Database (BSD) [46]. This database, at the time of writing, contained 500 images spanning a wide range of scenes. Each image has a number of ground truths associated with it as shown in Figure 3.2. The ground truths are referred to in this thesis by the index they are stored in by the database.

A ground truth for this database is a segmentation determined by a human. These ground truths, after being created, were associated with each image in no particular order and with no consistent properties. The result of this method of association is that the properties of

the ground truths associated with a particular index are essentially random.

The fact that this database is a segmentation database is not a problem. A segmentation, in a broad sense, is edge detection of only the most important edges in a scene. Due to this, the database gives a strict standard upon which to base the experiment if the interest is in an *edge image* with only the most important edges given.

Figure 3.2 Example of Berkeley Data Set Image with Ground Truth, Snow Shoes (2018)

Coupled with this database is an error metric needed to compare a potential *edge image* to the ground truths contained in the database. The error metric to be used for both systems implemented in this thesis is the BDM which was outlined in Section 2.1.2.

3.2 Image Feature Extraction

The premise of this thesis is that edge detection can be significantly improved by associating an appropriate edge filter with an input image. Hence, a method for associating an incoming image to the proper filter to solve its *edge image* is needed. This problem is a variation of the scene recognition problem. Typically, in a scene recognition request there are a predetermined set of classes which the image is supposed to map to. In addition, there is some a priori information about the scene which can be used for feature selection. The challenge with this scene recognition problem is there is no way of knowing what constitutes a different scene as the *scene* is defined as an image which is solved by a particular filter. Thus scene recognition approaches which rely on object information or regions of interest will not be useful. The technique used needs to operate without the benefit of any a priori information. A GIST approach to scene recognition will be used.

It has been shown by Siagian and Itti that the usage of a GIST feature set results in favorable scene recognition for learned outdoor scene and scene segments [5]. The feature set is constructed from orientation, color and intensity information as given by Figure 3.3. The first step of feature extraction was to create an eight layer spatial pyramid by successively applying a Gaussian filter and down sampling by 2. This creates a measure of scale invariance for the feature set. Four orientation channels were created using Garbor filters at: 0, 45, 90 and 135 degrees. Each channel was processed at each layer of the spatial pyramid.

Color features were created using opponent colors and intensity. From a RGB image the opponent colors red, green, blue, and yellow as well as intensity are extracted by using

equations 3.1-3.5:

$$R = r - (g + b)/2 \tag{3.1}$$

$$G = g - (r + b)/2 \tag{3.2}$$

$$B = b - (r + g)/2 \tag{3.3}$$

$$Y = r + g - 2(|r - g| + b) \tag{3.4}$$

$$I = (r + g + b)/3 \tag{3.5}$$

The color feature images have a center surround operation which is similar to a difference of Gaussian operation performed on them. This operation subtracts an up sampled image from lower on the pyramid from an image higher on the pyramid.

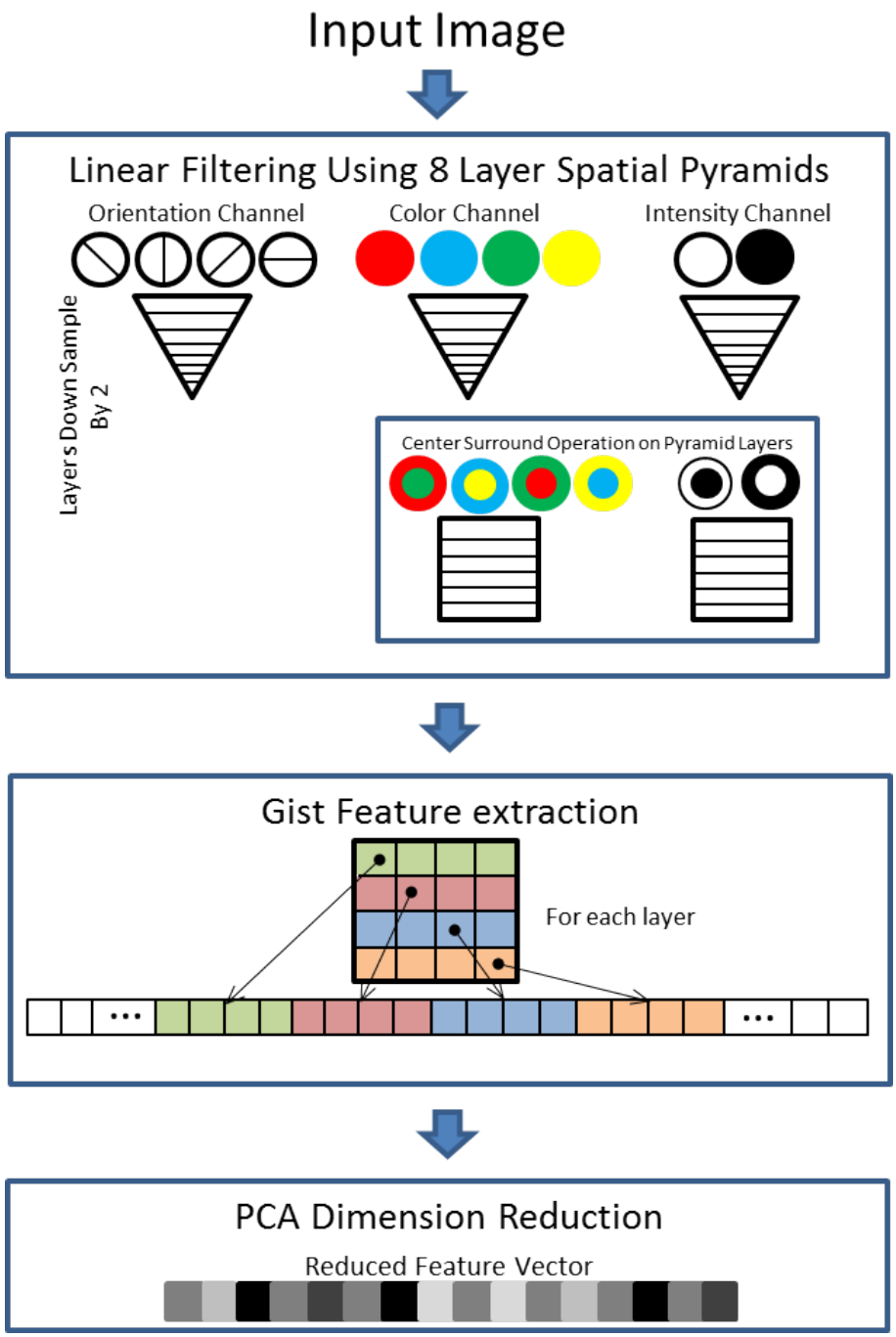
$$Featureimage = P(h) - upsampled(l - h)(P(l)) \tag{3.6}$$

Where P is the spatial pyramid being worked with, h is the layer of the higher resolution image and l is the layer of the lower resolution image. The result of this operation is a comparison between the center pixels and its surrounding neighbors. For each pyramid, the following layer combinations were used for each opponent color pair: layer 2 with layer 5, 2 with 6, 3 with 6, 3 with 7, 4 with 7, and 4 with 8.

Each feature image which was created, 34 in total, was then subdivided into 16 equal regions. Each region is $\frac{1}{4}$ the height and $\frac{1}{4}$ the width of the original image. The mean of each region is taken. These means when concatenated together into a single vector are the GIST vector for that image.

The resulting GIST feature vector is $32 * 16 = 544$ elements in length. There are many redundancies in this data set. Siagian and Itti reduced the feature set using PCA and ICA [5]. It was determined that the inclusion of ICA did not yield a significant improvement of output over using the much faster PCA algorithm alone. As such, the feature set is reduced using only a PCA algorithm. The feature set will typically reduce to 80 features, a much more manageable number while still maintaining 97% of the information.

Figure 3.3 Three Feature Channel Scene Classification [5]



3.3 Edge Detection Filter Creation

The input to the system, the image features, were discussed above. The proposed output of a system built from the proposed framework are edge filters, or edge detection methodologies. The training set for the input is created by taking the training images and extracting the features from them. The training set for the output is created by taking the images and determining the best filters or methodologies in the chosen search space for that image. Those methodologies are then associated with the training input to create a training set. The set of all filters in the training set are the training filters. For the prototype system a filter generator based on CA will be used.

Uguz et al. proposes two methods of edge detection using CA which give better performance than standard Prewitt, LoG, Roberts, Canny and Sobel methods in many cases [12] [18]. Uguz et al. demonstrates that simply applying a linear CA rule to a binary image can produce results that are on the whole superior to both the Canny and Sobel methods of edge detection, given the proper CA rule is chosen [12]. Uguz et al. later augments this approach by using PSO to train fuzzy CA filters for edge detection [18]. While these methodology do produce impressive edge results, there is a price; both methods require a ground truth to train against and they currently have no way to generalize. In order for these methods of edge detection to be feasible for mobile robotics applications, the need of a ground truth for every input image must be removed. These two methodologies given in [12] and [18] will be used as the edge detection filter creators for this research.

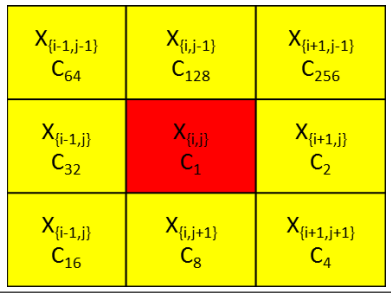
The remainder of this section will explain how these methodologies function and are adapted to work with the overall system. First the plain Linear CA filter, or Non-Fuzzy

approach, will be explained. Then the process by which a Fuzzy CA filter set is created will be introduced.

3.3.1 Linear Cellular Automata

As mentioned in section 3.3, the search space of all CA transition rules is exceptionally large. With this in mind and in the interest of achieving an efficient solution Uguz et al. confined their filter search to the set of Linear CA [12]. This set is built out of a 3x3 pixel neighborhood as given by Section 3.4 and Figure 3.4. The different automata are created by including or excluding different pixels from the neighborhood. This is indicated by the pixel in the neighborhood mask being a 0 for exclusion or 1 for inclusion. The automata are indexed by the binary representation of their included and excluded cells as Rules 1 through 512. The pixel location is updated using the transition function given by equation 3.7.

Figure 3.4 Linear CA Neighborhood Inclusion Map



$$x_{i,j}^{t+1} = (c_1 x_{i,j}^t + c_2 x_{i,j+1}^t + c_4 x_{i+1,j+1}^t + c_8 x_{i+1,j}^t + c_{16} x_{i+1,j-1}^t + c_{32} x_{i,j-1}^t + c_{64} x_{i-1,j-1}^t + c_{128} x_{i-1,j}^t + c_{256} x_{i-1,j+1}^t)(mod2) \quad (3.7)$$

- c is the inclusion variable, 1 indicates that that neighborhood pixel is used in determining the next state, 0 indicates that it is not.
- $x_{i,j}$ indicates a pixel location
- t is the time step

Uguz et al. manually searched through all of the linear CA rules to find the one that would yield the best *edge image* for an input image [12]. This was done for three very common test images: *Camera Man*, *Lena*, and *Peppers*. This hand selection yielded some excellent *edge images* with a much smaller computation cost then standard methods. It is not practical for this prototype system to hand select the filter that is to be used; this would be too time consuming. More importantly it would harm the modular nature of the overall system. It is imperative that the system be autonomous after the initial defining of the desired ground truth. To that end, a brute force approach was taken to determining the best filter for the Non-fuzzy linear CA approach. Because the filter space was reduced to a small 512 possibilities it is simple to apply each CA filter to each image and then use the BDM to gauge how well the filter performed. The filter with the lowest BDM is then determined to be the best filter for solving that particular training image and is associated with that image as its filter for training. When the filter space is expanded to the whole of the CA filter space this approach will be impractical and a learning approach such as the one given in Section 3.3.2 will need to be employed.

3.3.2 Fuzzy Cellular Automata

Uguz et al. extends the research done on simple linear CA rules to include fuzzy logic rules [18]. This addition is important because it allows the CA filters to operate on grey scale images as opposed to only monochromatic images. The inclusion of this added information in the *edge image* creation greatly increased the performance of the CA Filters while simultaneously increasing the possible filter space. This increase in filter space requires that a learning algorithm be used to search the space rather than an exhaustive search being performed.

The fuzzy transition rule used by Uguz et al. is a near neighborhood CA as given by equation 3.8.

$$F(C_{i,j}) = \begin{cases} 1 & ,\mu(C_{i,j}) > \tau \\ 0 & ,\mu(C_{i,j}) \leq \tau \end{cases} \tag{3.8}$$

- $C_{i,j}$ is the central pixel in a Moore neighborhood CA
- $F(C_{i,j})$ is the fuzzy transition rule
- $\mu(C_{i,j})$ is the degree of edginess of each pixel

$$\begin{aligned} \mu(C_{i,j}) &= \frac{\Phi(C_{i,j})}{\Delta + \Phi(C_{i,j})} \\ \Phi(C_{i,j}) &= \sum_k \sum_l |C_{i,j} - C_{i+k,j+l}| \end{aligned} \tag{3.9}$$

  where $k, l \in \{-1, 0, 1\}$

- τ is the edge pixel Threshold

A PSO algorithm was used to train a CA edge detection filter to each image in the training portion of the BSD. Only one of the ground truths associated with each image was used for training. The PSO is initialized with 20 particles each with a location which consists of a fuzzy offset parameter, a fuzzy transition rule boundary τ, and a linear CA rule. The location and the rate of change of the location, the velocity, are updated every time step using equations 3.10 and 3.11 [18]. The same velocity is applied to each coordinate in the position.

$$x_k(t+1) \quad = v_k(t+1) + x_k(t) \tag{3.10}$$

$$\begin{aligned} v_k(t+1) \quad &= w * [v_k(t) + r_1 * c_{1k} * (p_{best_k} - x_k(t)) \\ &+ r_2 * c_{2k} * (g_{best_k} - x_k(t))] \end{aligned} \tag{3.11}$$

$$w \quad = \frac{2}{|2 - \phi - \sqrt{\phi_2 - 4\phi}|}$$

- $x_k(t)$ is the position at time t
- $v_k(t)$ is the velocity of the particle at time t
- c_{1k} is the particle history factor
- c_{2k} is the swarm influence factor
- r_1 and r_2 are uniform random variables $\in \{-1, 0, 1\}$ used for scaling
- $\phi = c_{1k} + c_{2k}$ given that $\phi > 4$

- w Is the constriction factor, used to control the learning factor and amplitude of the particles oscillation. This value is set to 0.7289
- p_{best_k} is the best recorded position of the particle.
- g_{best_k} is the position of the particle which has the best overall location.

In the Non-Fuzzy method, each image was trained separately to a filter that would best solve the image according the BDM. This was an excellent strategy based on the planned training method where the output of the network was more akin to a regression than a strict classification. The output of the other network represented every possible CA filter. This is not possible for the Fuzzy approach which has a truly infinite number of combinations due to the continuous nature of two of its variables (the fuzzy boundary and the offset). Thus, it is not practical to train each image to a specific filter because that will, in all likelihood, yield all unique filters which have little in common. This would add nothing to the systems ability to handle unknown images. Instead, a boosting approach was adopted.

As stated in Section 2.3.4, boosting generates a collection of weak learners that together form a strong learner. In a similar manner, this system will generate a collection of weak filters, filters which only solve a certain class of images, which when taken together with the rest of the system result in a strong edge detection method. In this instance, the classes will be *solves the edge detection satisfactorily* and *does not solve the edge detection satisfactorily*. The error of an *edge image* is calculated using the BDM. An error threshold is determined by the user for an individual *edge image* solution as well as for the entire set. For this case, the error threshold is taken to be the lowest error on the image generated by

the standard edge detection methods 2.1.3.

While an acceptable edge solution has not been found, for each *edge image* a new filter is trained against an image which has no filters associated with it. The new filter is then applied to all images. If, when applied to an image, the filter produces an acceptable *edge image* (one with a BDM lower than the standard methods) the filter is added to that images list of filters. The end result is a list of images each with a list of filters which solve them better than the standard methods. This is a many to many relationship from filters to images. In order to train a network there needs to be a one to many relationship from filters to images. This means that each image can only be related to one filter, but each filter may be related to more than one image.

The reduction is accomplished by identifying the filters which are most likely to best accomplish the end goal of the system, creating a robust edge detection system. Thus, the best filters to keep would be the filters which have the smallest BDM while also solving the largest number of images. First the number of images that each filter solves is tallied. The filter that solves the largest number of images is added to the set of final filters. The Images that are solved using that filter are eliminated from the tally. This process is repeated until every image has been eliminated. The final result is the smallest set of filters which will solve all of the images while maintaining a BDM lower than the standard methods.

3.4 Filter Selection

The final module of the framework teaches a system to associate incoming images with the filter that will solve for the edges of that image. There are a number of methodologies that will serve the purpose of this section. Any classification network will suffice. What

must be kept in mind is the end goal that the network is training towards and then setting up the network appropriately to accomplish that end. For the prototype systems a back propagation neural network was used to train each image to its filter. Neural networks were chosen because of their success in performing scene recognition tasks as seen in section 2.1.8. The neural network uses the filters to learn the classes of the images. The classes of the images are hidden inside the network themselves. It is this image class knowledge that will allow the networks to identify the appropriate filter to use.

3.4.1 Linear CA Neural Network

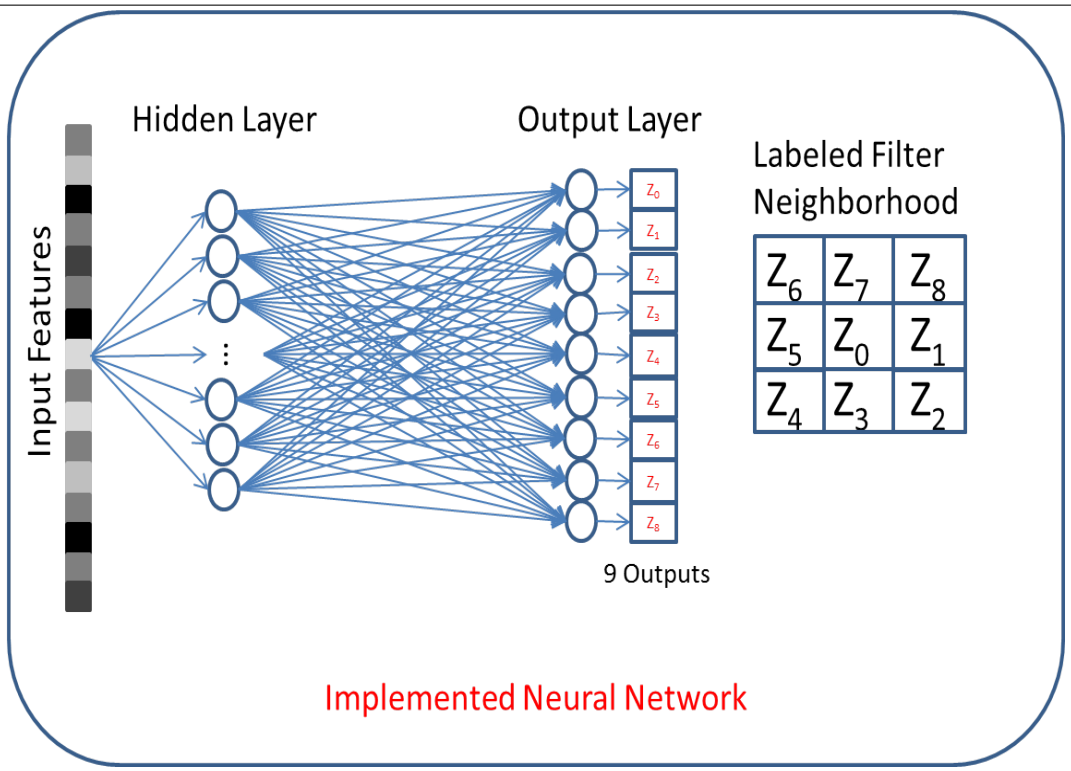
For a stand alone Linear CA filter, the only information needed to distinguish between filters is which pixels in its neighborhood are to be used to determine its next state. In this system, the next state determines if the pixel of interest is an edge pixel. This system uses a Moore neighborhood, meaning there are 9 pixels which are either included or excluded from the neighborhood. Thus, the output of the network need only be 9 nodes, one for each neighborhood location. The output will be interpreted as a binary encoded network. If a pixel is to be included, the output location corresponding to it will have a 1 otherwise a 0. By binary encoding the network, the system will be able to assign any filter in the filter space to the input image. This gives the network some regression capabilities as opposed to being a strict classifier. Thus, this network will be able to handle input images which it was not trained against.

The input to this network will be the features gathered according to Section 2.1.8. The features are given to the network as a column vector. In the event that multiple images are considered at the same time, each column is a new sample. Due to the small number of

training samples available, re-sampling will be done to increase the training set to a number which is ten times larger than the number of weights in the network. This allows for proper network training. Only a single hidden layer will be used for the prototype system. Various numbers of nodes will be used to augment the size of the hidden layer of the network.

The end result of this training will be a neural network which takes in a new image and then outputs a new filter based on the features of the image, Figure 3.5.

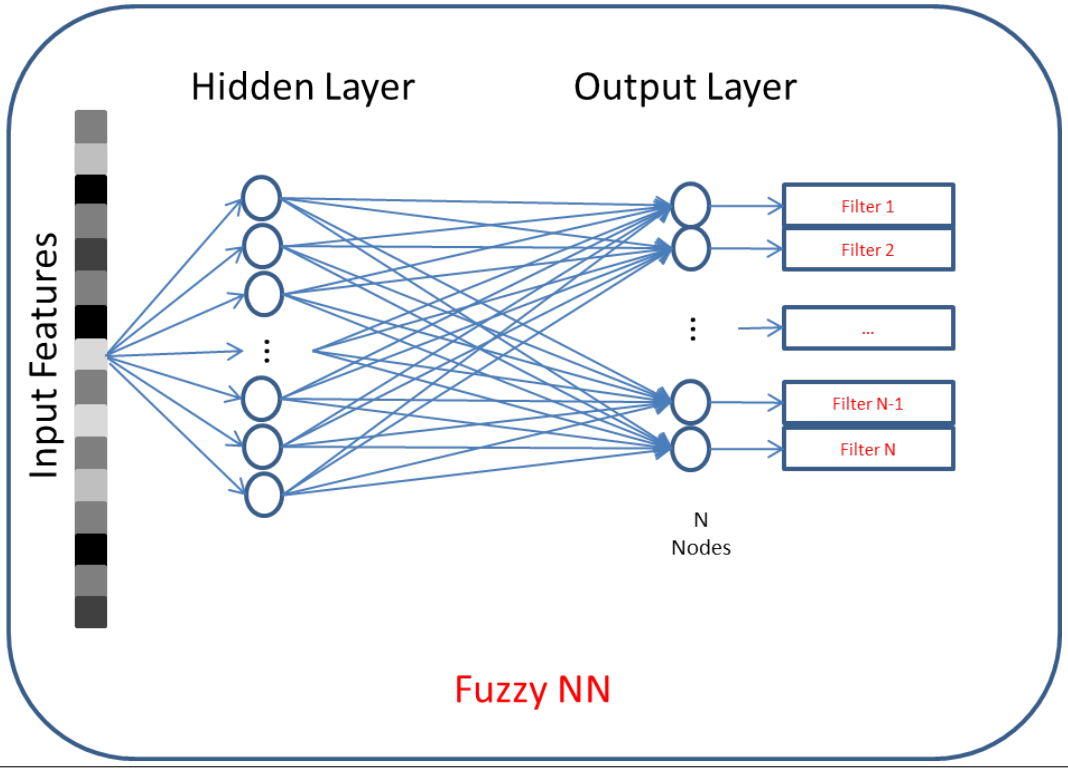
Figure 3.5 Non-Fuzzy CA Neural Network



3.4.2 Fuzzy CA Neural Network

The Fuzzy CA NN will also use the features gathered according to 3.2 as well as only a single hidden layer tested with various numbers of nodes. The difference is in what the neural network is expected to output. In this case the output will not reference the entire filter space. This would not be possible as outlined in Section 3.3.2. Instead, this network will be a true classifier. Each output will correspond to a single filter as given in Figure 3.6. Thus, this method will select from a filter bank as opposed to creating a new filter for every input image.

Figure 3.6 Fuzzy CA Neural Network



Chapter 4

Results

This chapter investigates the performance of the implemented prototypes of the proposed framework. Many aspects of the systems will be investigated:

- The type of ground truth the system performs well on, if any.
- The percentage of images on which the system outperforms all the standard methods.
- The size of neural network needed to attain the best results.
- The number and type of filter which is selected.
- The type of image that the system performs well on, if any.

All of these aspects will give insight into the workings of the framework and the specific implementation chosen. In addition, the aspects will reveal potential ways to improve future systems. In order to determine the success of the system two tests will be used.

- The first test will be done on a set of 100 images with ground truth from the Berkeley Segmentation Database which the system has never seen before. The test will be considered passed if the implemented systems can outperform the standard methods in cumulative BDM as well as number of images where the prototype system performs the best.

- The second test will be done on images from the MIT Sun Database [47]. This database does not have any ground truths for *edge images* so the comparison for successful edge detection from these images will be purely subjective. Thus, success of the system will also be subjective based on this test. This test is important because an edge detection methodology is only useful if the edges are acceptable to a human user.

If either system can pass both tests, then the framework is proven to be a benefit for edge detection applications where the scene is not known before hand.

4.1 Test On Image Database With Ground Truth

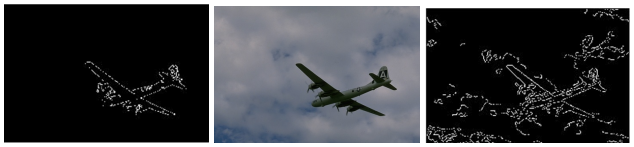
The prototype systems of the framework are trained in two parts. First, the set of filters are trained to the individual training images. Then the neural network which pairs an image with its filter is trained. In order to facilitate investigation of these two parts, a single set of filters is trained and then used many times to train the neural network. Throughout the tests, these filter sets will be referred to as runs. Within each run, are a set of filters is trained against a specific set of ground truths. The ground truths sets are referred to by their index as given by the BSD database. Thus, when referring to a system its run and ground truth are given to uniquely identify the filter set to allow for replication of results. When the filters from a run are used to train a neural network, the unique properties of the neural network will be used to identify the network, and hence uniquely identify the system.

4.1.1 First Filter Set for Non-Fuzzy Prototype System

An initial run of the system was taken using only the simple Non-Fuzzy CA implementation. The purpose of this run was to see how the system performed in a general sense and investigate ways to better configure both Fuzzy and Non-Fuzzy solutions.

Five filter sets were trained each against the same ground truth index of 400 training images. These five networks were then each trained against all five ground truths and tested against 100 new testing images and ground truths of the same index. To illustrate the outcome of the prototype system a representative test image was selected with its accompanying Canny and Sobel *edge images*, Figure 4.1. The ground truths which accompany this image are given by, Figure 4.2.

Figure 4.1 Airplane (3096)



(a) Sobel (b) Original Image (c) Canny

Figure 4.2 Airplane (3096): Ground Truths Index 1-5

(a) Ground Truth 1 (b) Ground Truth 2 (c) Ground Truth 3 (d) Ground Truth 4 (e) Ground Truth 5

The first thing to notice is the great difference in the edge interpretations between the ground truths. Indexes 1, 4 and 5 are only interested in the plane to various degrees of detail while 2 and 3 also outline the clouds. Figure 4.3 gives the *edge images* produced by the five systems and Table 4.1 gives the corresponding performance information. For

each system, the filter selected was different but the *edge image* was very similar. This indicates that there are several CA which will behave in a similar manner for this image. The same phenomenon is also observed in many of the other test images. This observation further indicates that the neural network was able to learn which CA would work well. By comparing the BDM across the different systems the impact of the various ground truths is seen. Though the *edge image* is similar the BDM varies widely. This is confirmed by looking at the Sobel and Canny BDM across the different ground truths because these *edge images* do not change.

For ground truth 2 and 3, which are interested in the plane and the sky, the system outperforms both Canny and Sobel methods as seen in Table 4.1. However, when only the plane is of interest the system only outperforms Sobel and not Canny. This is apparent when comparing the *edge images* from Figure 4.1 and 4.3. It is observed that Canny has the largest amount of detail hence it will only perform well when many edge pixels are present in the ground truth. In contrast, Sobel only captured the plane and hence had fewer edge pixels so it will perform better then Canny when there are fewer ground truth edge pixels. This was seen on many of the other test images. The prototype system performs in between Canny and Sobel in number of edge pixel detected. Hence, it performs well on the ground truths requiring a higher number of edge pixels.

These trends are fairly consistent across all test images as illustrated by Table 4.2 where the sum of the BDM across all test images is given. The system consistently outperforms Canny because Canny consistently has more false positive edge pixels. However, the system is consistently outperformed by Sobel for the same reason. Many additional runs and

configurations were attempted for comparison but all with similar results.

Table 4.1 Airplane (3096): BDM Error from System Trained and Tested Against Same Ground Truth Index, Neural Network Hidden Layer 100 Nodes

Ground Truth	Test	CA Filter	Sobel	Canny
1	88.6607	404	**45.142**	115.9672
2	**20.2267**	118	66.912	47.5669
3	**21.9209**	212	64.911	45.104
4	88.3304	295	**44.8705**	115.9386
5	88.642	248	**44.9191**	116.0968

Figure 4.3 Airplane (3906): *Edge images* Selected by Five Different Prototype Networks

(a) Network 1 (b) Network 2 (c) Network 3 (d) Network 4 (e) Network 5

Table 4.2 Cumulative BDM Error: System Trained and Tested Against Same Ground Truth Index, Neural Network Hidden Layer 100 Nodes

Ground Truth	Test	Sobel	Canny	% Outperform
1	5532.057	5150.502	5853.591	19
2	5317.879	5024.691	5615.897	19
3	5281.602	4890.233	5681.595	20
4	5334.289	4873.972	5704.957	19
5	5426.856	4892.934	5791.787	15

The prototype system was also used to investigated the significance of different numbers of nodes in the hidden layer of the neural network for training. Table 4.3 indicates that there is little difference between the number of nodes used on average across all images in the test set. Thus, for better performance and less risk of over training a smaller network should be used.

A sampling of successful *edge images* was taken to investigate the type of image that this system performs well on. It can be seen from Figure 4.4 that the system performs well on a wide cross-section of images. 'Couple' and 'Worker' are both of people on

Table 4.3 Cumulative BDM Error:System Trained and Tested Against Same Ground Truth Index, Neural Network Hidden Layer 25, 50, and 100 Nodes

# Nodes	Test	Sobel	Canny	% Outperform
25	5522.686	5150.502	5853.591	19
50	5522.318	5150.502	5853.591	19
100	5522.116	5150.502	5853.591	19

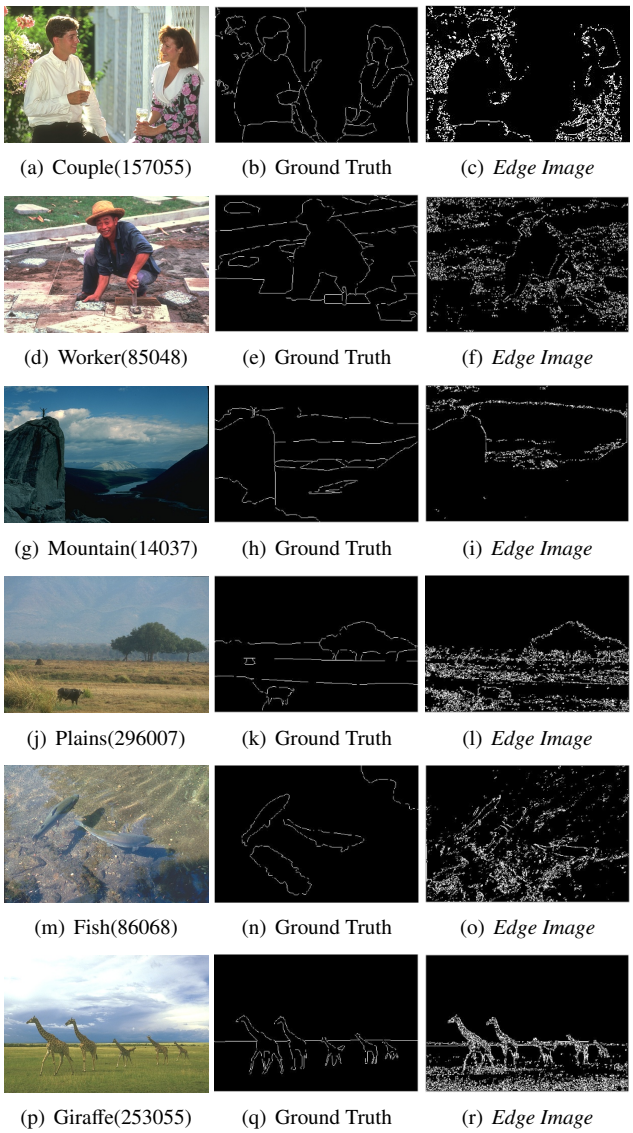
complex backgrounds, 'Mountain' and 'Plains' are of different natural scenes, and 'Fish' and 'Giraffe' are both of different types of animals in their natural environment. In each case, the system selected a filter which performed better than both Canny and Sobel. The degree to which the performance was better can be seen in Table 4.4. This table also gives the filters which were selected by the system. None of the images were solved by the same filter. This shows that the network is intelligently selecting the filters. This indicates that that the scene and edge information are related, and more importantly that the relationship is being learned by the neural network.

Table 4.4 Performance of Images in Figure 4.4 using BDM

Image	Ground Truth	CA Filter	Test	Sobel	Canny
157055	1	94	**15.677**	20.2186	28.3448
14037	3	23	**31.7453**	36.3771	40.4594
85048	1	14	**30.9666**	33.1152	35.8777
86068	1	19	**72.4349**	74.5816	79.4239
253055	3	279	**25.3907**	68.083	98.0544
296007	5	31	**24.5686**	35.7921	40.3939

In order to gain even further insight into the function of the trial system, the set of the successful images was analyzed across multiple runs of the same system. These results are given in Table A.1. As already established, the neural network is a trained network and hence every time that the neural network is trained the weights will very and hence the output may change. What is interesting here is that for many of the images multiple different filters were selected across the different networks. However, each of these selected

Figure 4.4 Set Of Images For Which The Protoype System Selected Filter Outperforms Standard Methods



(a) Couple(157055) (b) Ground Truth (c) *Edge Image*

(d) Worker(85048) (e) Ground Truth (f) *Edge Image*

(g) Mountain(14037) (h) Ground Truth (i) *Edge Image*

(j) Plains(296007) (k) Ground Truth (l) *Edge Image*

(m) Fish(86068) (n) Ground Truth (o) *Edge Image*

(p) Giraffe(253055) (q) Ground Truth (r) *Edge Image*

filters was able to outperform all of the standard methods. This shows that even in this small filter space there are a great number of filter solutions and they are not unique with regards to outperforming the standard methods. In addition, the images which the system outperformed on were the same for every case. As mentioned above, there is no consistent type of image which the system does better on then others. This indicates that the system

can be improved to do well on a greater portion of the images if not all of the images. A full list of images which the system performed well on, with accompanying images, is given in Appendix A.

4.1.2 Test of Revised Prototype Non-Fuzzy System and Fuzzy Systems

It has been shown that which ground truth index is used makes no difference. Thus, only the results from training the system using the first ground truth index will be presented. Both Non-Fuzzy and Fuzzy System were run. In addition to Sobel and Canny all of the other standard methods ,Prewitt Roberts and LoG, are tested against to show the prototype system's relative performance.

Six networks were trained against the first ground truth index. The cumulative BDM for each network is given in Table 4.5. Three of the networks were created using the Non-Fuzzy Systems and the other three using the Fuzzy system. Each type of system trained a network that has 25, 50, and 100 nodes in its hidden layer. It can be seen in both systems that the size of the neural network made little difference. Only a slight improvement was observed in the smaller network over the larger networks. This improvement is related to an over-training problem observed in both networks.

Even at a network size of 25 nodes over training is observed. Figures 4.5 and 4.6 give the confusion matrix for the resulting error of the Non-Fuzzy and Fuzzy systems respectively. The Non-Fuzzy network essentially has 512 classes. The training observed is 82.3% correct which is unexpectedly high. Likewise, for the Fuzzy system of 16 classes classifying to an accuracy of 99.8% correct is unheard of. This indicates that the scene to filter classification is highly separable, or that there is a large amount of noise in the system which is being

learned. Images are noisy systems which makes the over training explanation the more likely interpretation. This explanation is supported by the performance of the filter when it is shown images whose ground truth it does not know. The successful training rate drops to the more expected 30%. This means that the noise was carried through when the system was applied to the unknown testing images, resulting in a higher number of misclassifications and hence overall lower performance. A smaller network should be used to avoid, as much as possible, the problems with over-training the system.

The most promising result in Table 4.5 is the 25 node fuzzy network. This network has a cumulative BDM of 5159.716 which is only 3% higher then the best performing method, Roberts, at 5008.882. the prototype system need perform only 3% better in order to outperform all of the methods.

Table 4.5 Second Run: Cumulative BDM

# Nodes	Non-Fuzzy Network	Fuzzy Network	Sobel	Canny	Prewitt	Roberts	LoG
25	5525.9	**5159.716**	5150.502	5853.591	5132.561	**5008.882**	5785.039
50	5522.318	5445.108	5150.502	5853.591	5132.561	5008.882	5785.039
100	5522.116	5313.69	5150.502	5853.591	5132.561	5008.882	5785.039

The 25 node neural networks are now the only networks to be considered for the rest of this section. Table 4.6 gives the number of images for which each method had the best BDM. By this metric, the prototype systems fall short of proving the framework a success. Both Fuzzy and Non-Fuzzy systems outperformed all of the other methods save Roberts for this metric. The Fuzzy method did improve over the Non-Fuzzy method by almost

Figure 4.5 Confusion Matrix from Non-Fuzzy 25 Node Network Training

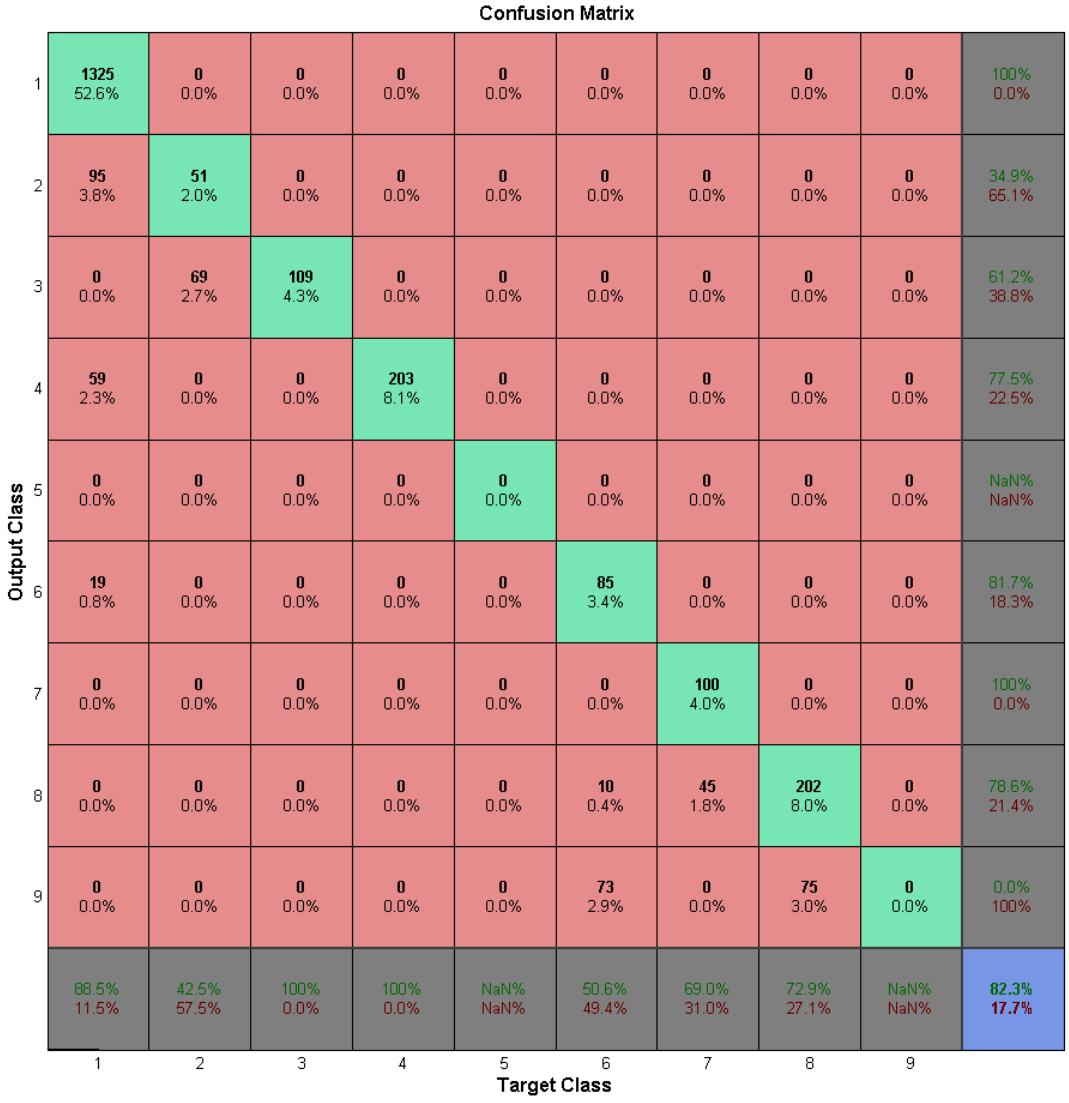
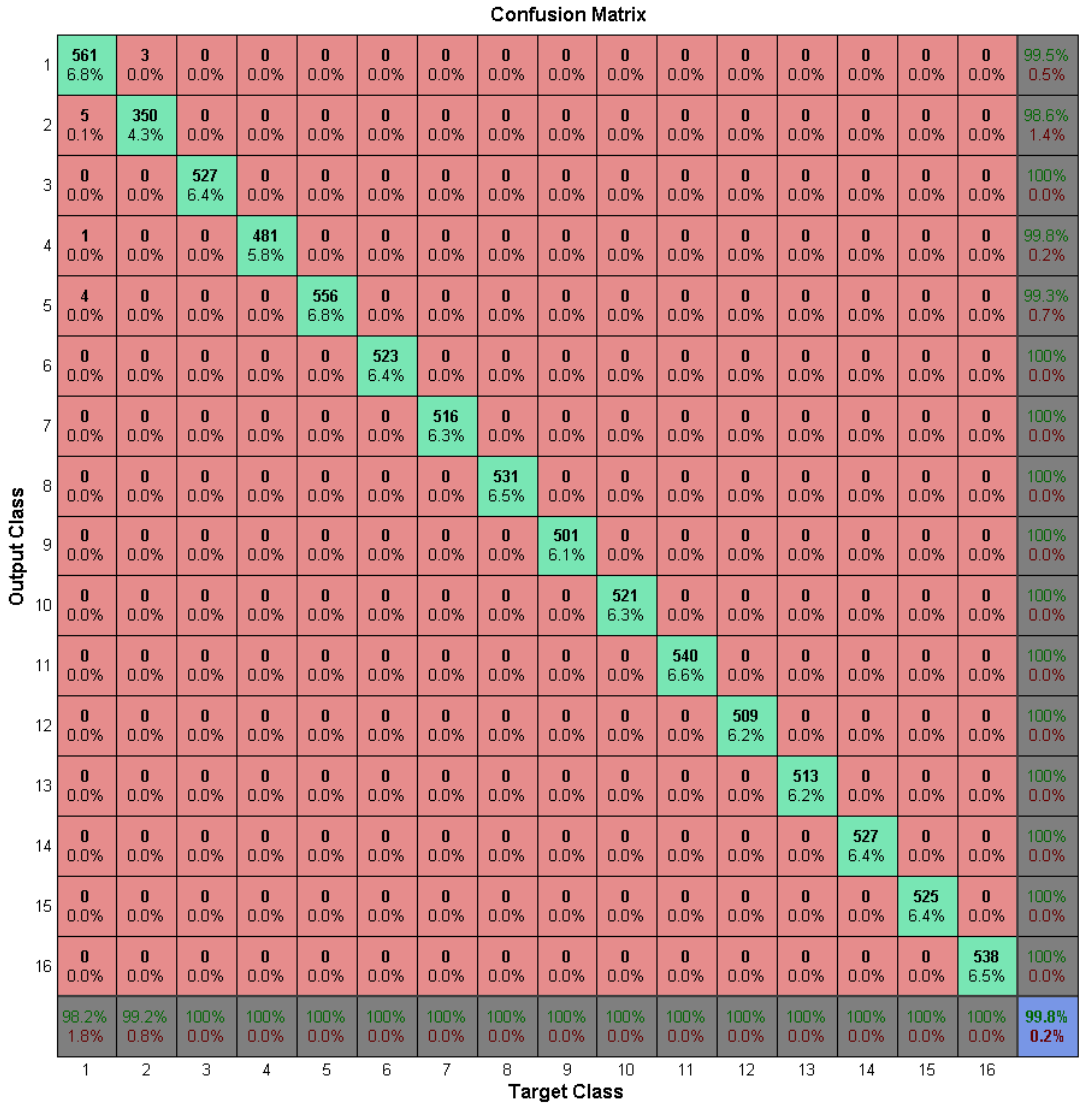


Figure 4.6 Confusion Matrix from Fuzzy 25 Node Network Training



doubling the number of images that it outperforms on. This was expected as the Non-Fuzzy method works on monochromatic images while the Fuzzy method is able to take advantage of more information by using grey-scale images. However, the Fuzzy method still solves 9 fewer images than Roberts. It is surprising that Roberts is the best method to use according to the metrics employed. It is commonly accepted that the Canny and LoG methods are the best to use in most edge detection applications because they give more complete edges. It is expected of the Roberts method to be slightly inferior but the fastest to implement in hardware. While neither of the prototype system's results clearly shows a numerical superiority to the standard methods, it does clearly indicate the Fuzzy methods superiority to the Non-Fuzzy according to the BDM.

Table 4.6 Second Run: Number of Images Each Method Performed Best On

System	# Nodes	Test	Sobel	Canny	Prewitt	Roberts	LoG
Non-Fuzzy	25	17	6	5	15	57	1
Fuzzy	25	31	7	5	15	40	2

A compelling reason for the poor behavior of the prototype systems is that they were trained on ground truths with essentially random properties. This means that there was not as consistent definition of an edge to train against. The next round of edges trained will be against ground truths which have the largest number of edge pixels available and the smallest number of edge pixels available. This increase in consistency of edge definition should yield an increase in the systems performance.

With the numerical results analyzed it is also important to investigate the quality of the results in a subjective manner with respect to the numerical results. A number of observations can be made from the images given in Figures 4.7, 4.8, 4.9, 4.10, and Table 4.7.

The first observation to make is that filters which create sparse *edge images* perform better then those which tend to produce busier *edge images*. This is the case because of the penalty for false positive pixels in the BDM. The penalty is assessed with respect to the distance the false positive is to the nearest ground truth pixel. This results in a large penalty for noise. This rewarding of sparse images leads to odd results such as the results of image in Figure 4.7 of three piglets. The Fuzzy system gave the best solution according to the BDM which is a nearly edgeless image that only captures a vague silhouette of the piglets. This image is missing a large number of edge pixels but it has very little noise resulting in a lower BDM than the other edges which capture all of the piglets but also have a large amount of noise. A small amount of noise far away from the central edges leads to a much higher BDM score than an empirical observation would give. In the case of the pig image, the Non-fuzzy system's *edge image* seems to have the best *edge image* though it had one of the higher BDMs.

Looking across all of the Canny and LoG *edge images*, it is apparent why the prototype system is able to out perform them consistently. Those methods, on their basic setting, have a tendency to produce a large amount of internal edges. In essence, these methods are prone to all of the noise in the test images. As a result, in many of the natural scenes, Canny and LoG perform poorly because of the textures present. A good example of this is given by the image of the giraffe in Figure 4.8. Sobel, Roberts, Prewitt and the prototype systems performed well. They have few edge pixels and those edge pixels are close to the ground truth edge pixels. Canny and LoG React to extraneous/noisy edges. The result is a large number of false positive edge pixels which decrease their performance.

The image of the kangaroo in Figure 4.9, though a failure of the system, lends a fair amount of insight into what is occurring. None of the methods are able to get a satisfactory *edge image*. This leads to the conclusion that the image is a challenging one to obtain an *edge image* from. This is caused by two factors. First the the green and brown color palette is fairly uniform which causes problems when using grey scale images for edge detection. The second is the rough texture of the image. The two prototype systems handled these problems in two different ways. The Non-Fuzzy system produces a busy *edge image* and the Fuzzy system produces a very sparse *edge image*. This shows the failure conditions of the prototype systems. The Fuzzy systems will tend to give blank *edge images* when no good filter match is possible based on the training set. This is a result of selecting a filter from a filter bank. An incorrect classification was made resulting in the selection of a filter with a fuzzy threshold which was too high. The Non-Fuzzy system is able to compensate better because it produces a filter rather than selecting one from a bank.

There are a number of images where the resulting *edge image* are very similar and yet the BDM score very different. A good example is image 4.10. Here the results from Sobel, Prewitt, Roberts and the prototype systems are very similar, and sparse when compared to the Canny and LoG *edge images*. Yet the Fuzzy system performs the worst. In this case the cause is the ground truth having a number of edge pixels spread out throughout the image. This allows for lower scores to be assigned to noise pixels because they are never far from an edge. Thus, in this case missing an edge pixel is worse then having a large number of noisy pixels. The Fuzzy method has fewer of the upper edge pixels and all of the same noise pixels, at the bottom of the image, as the rest of the methods. Canny and LoG have

the upper edge pixels plus upper noise which is not as large of a problem in this case. To solve this discrepancy in evaluation an *edge image* characterization method which limits the number of test edge pixels which can be mapped to the GT edge pixels would need to be used.

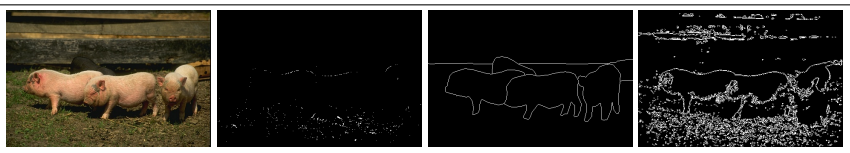
How well an *edge image* does is still very subjective even when a ground truth is used. Some people would prefer to have more edges, others less, some filled in, others not. The important thing for this research is to show that the system is able to be trained to meet the edge definition chosen across a wide range of images.

Table 4.7 Example Images: Filter, BDM and Standard Method Comparison Data

Image	Fuzzy-Filter	Fuzzy BDM	CA Filter	CA BDM	Sobel	Canny	Prewitt	Roberts	LoG
66053	82 , 0.712 , 178	*37.687*	294	*65.027*	64.588	69.003	64.794	44.601	68.859
253055	82 , 0.712 , 178	17.777	23	25.365	68.003	97.903	67.643	55.188	95.859
69020	124 , 0.67 , 203	78.956	456	67.793	62.839	67.007	62.62	60.997	66.731
41033	35 , 0.71 , 45	*62.675*	23	*42.117*	44.81	52.721	44.632	45.259	52.152

Of all the images which the systems solved individually, there were only four which they both solved better than the standard methods. Table 4.8 gives the numerical results of these images. Though the fuzzy system performs better overall, it only performs better than

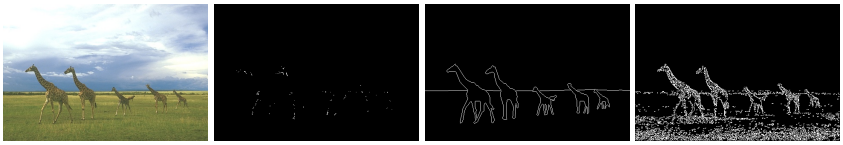
Figure 4.7 Piglet (66053)



(a) Original (b) Fuzzy (c) GT (d) Non-Fuzzy

(e) Sobel (f) Canny (g) Prewitt (h) Roberts (i) LoG

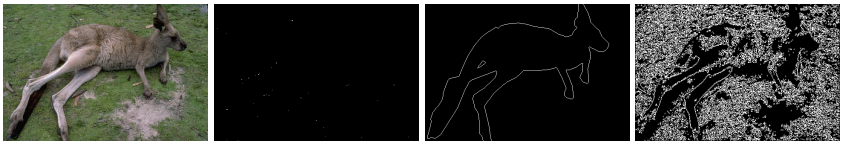
Figure 4.8 Giraffe (253055)



(a) Original (b) Fuzzy (c) GT (d) Non-Fuzzy

(e) Sobel (f) Canny (g) Prewitt (h) Roberts (i) LoG

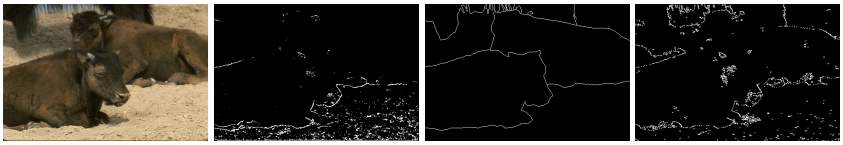
Figure 4.9 Kangaroo (69020)



(a) Original (b) Fuzzy (c) GT (d) Non-Fuzzy

(e) Sobel (f) Canny (g) Prewitt (h) Roberts (i) LoG

Figure 4.10 Cow (41033)



(a) Original (b) Fuzzy (c) GT (d) Non-Fuzzy

(e) Sobel (f) Canny (g) Prewitt (h) Roberts (i) LoG

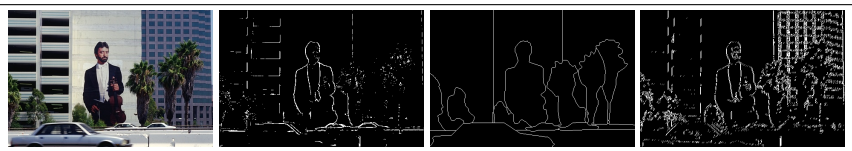
the Non-Fuzzy system on one of the four images below, the Billboard. These numerical results are supported by the subjective evaluation as well.

Table 4.8 Images Both Prototype Systems Solve Best: Filter, BDM and Standard Method Comparison Data

Image	Fuzzy-Filter	Fuzzy BDM	CA Filter	CA BDM	Sobel	Canny	Prewitt	Roberts	LoG
119082	82 , 0.712 , 178	**31.485**	409	34.378	36.571	40.946	36.642	36.505	40.346
157055	35 , 0.71 , 45	18.508	94	**15.677**	20.219	28.345	20.695	19.482	27.397
167062	82 , 0.712 , 178	93.829	246	**88.315**	111.88	122.54	111.42	114.17	121.49
42012	92 , 0.6 , 326	32.117	464	**26.862**	33.773	32.947	33.422	33.35	33.603

A subjective evaluation of the successful images was also conducted. The images for this evaluation can be found in Appendix B. It was observed that in general the Fuzzy systems *edge images* appear to be the best.

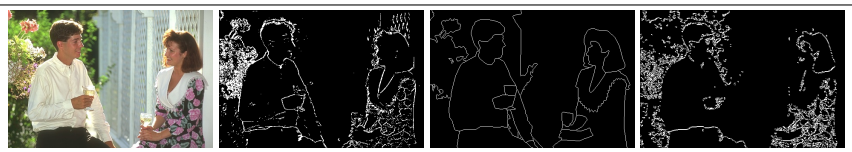
Figure 4.11 BillBoard (119082)



(a) Original (b) Fuzzy (c) GT (d) Non-Fuzzy

(e) Sobel (f) Canny (g) Prewitt (h) Roberts (i) LoG

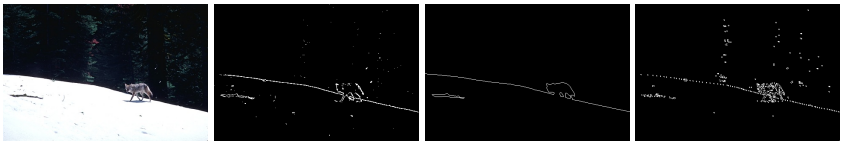
Figure 4.12 Couple (157055)



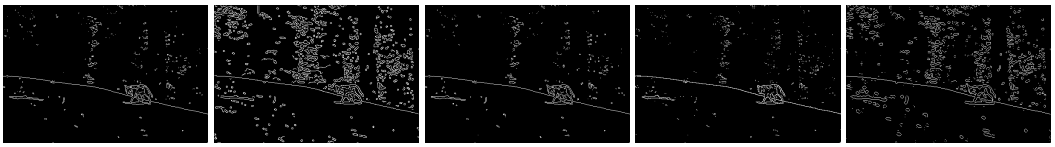
(a) Original (b) Fuzzy (c) GT (d) Non-Fuzzy

(e) Sobel (f) Canny (g) Prewitt (h) Roberts (i) LoG

Figure 4.13 Wolf (167062)



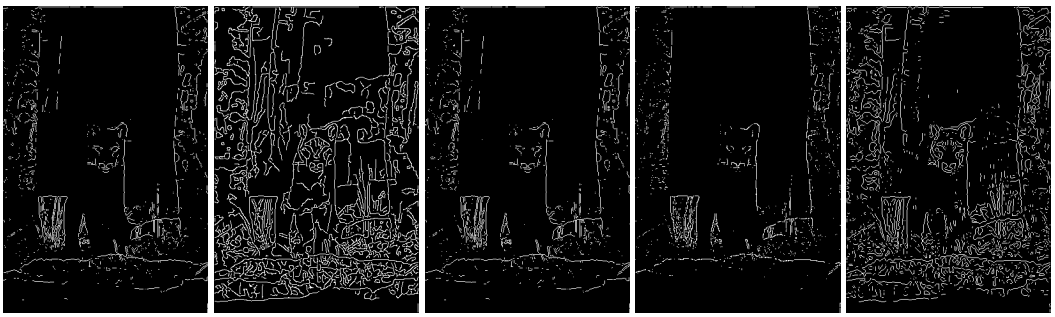
(a) Original (b) Fuzzy (c) GT (d) Non-Fuzzy



(e) Sobel (f) Canny (g) Prewitt (h) Roberts (i) LoG

Figure 4.14 Cougar (42012)

(a) Original (b) Fuzzy (c) GT (d) Non-Fuzzy



(e) Sobel (f) Canny (g) Prewitt (h) Roberts (i) LoG

4.1.3 Test of Prototype Systems on Busy and Sparse Ground Truth *Edge Image*

The results from Section 4.1.2 made it clear that a better system would result from creating filters and training the system against ground truths which had a more unified definition for an edge. The feature which appeared to have the most significance in the edge definition was the number of edge pixels present in the ground truth. The ground truths which had fewer edge pixels were producing very different results from the ground truths which had many edge pixels. It seemed reasonable that a system designed for edge detection should have at least a loosely unified definition of an edge. For the third run the systems were trained for situations where there were the largest number of edge pixels available in a ground truth, a busy *edge image*, and the smallest number of edge pixels, a sparse *edge image*. It is to be remembered that this is a segmentation database and, as such, none of the ground truths will have an exceptionally large number of edge pixels. In total 4 systems are considered in this run: a sparse ground truth Non-Fuzzy system, a busy ground truth Non-Fuzzy system, a sparse ground truth Fuzzy system, a busy ground truth Fuzzy system.

Tables 4.9 and 4.10 give the cumulative results for the four systems considered. In addition another method of standard edge detection is compared against, a method which uses color vector fields to determine edges, see section **??**. The threshold is set to .7 out of 1. This method was chosen to show that according to the BDM these prototype systems could even outperform a method using color information across a wide range of scenes. It was thought that the prototype system would outperform the standard methods when using the busy ground truths and would not when using the sparse ground truths. This was not the case. The cumulative BDM in Table 4.10 shows that the busy systems still

performed worse than many of the standard methods, however the performance improved dramatically for all the systems relative to the results for the second run. In contrast, the sparse systems cumulative BDM was worse than the results from the second run across the board. However, the Sparse Fuzzy system outperformed all the standard methods, including the CVF, in both BDM and number of images solved best.

Table 4.9 Comparison by Number of Images Each Method Performed Best On, 25 Hidden Layer Node Neural Network

System	GT	Test	Sobel	Canny	Prewitt	Roberts	LoG	Color Vector Field
Non-Fuzzy	Sparse	16	3	0	6	66	1	8
Non-Fuzzy	Busy	15	6	8	18	45	8	0
Fuzzy	Sparse	**49**	**4**	**1**	**4**	**30**	**1**	**11**
Fuzzy	Busy	12	7	11	15	37	9	9

Table 4.10 Comparison by Cumulative BDM, 25 Hidden Layer Node Neural Network

System	GT	Test	Sobel	Canny	Prewitt	Roberts	LoG	CVF
Non-Fuzzy	Sparse	6931.8	6448.53	7431.07	6431.81	6236.13	7266.47	6735.94
Non-Fuzzy	Busy	4198	3895	4312.1	3875	3911.2	4243.2	4061.77
Fuzzy	Sparse	**6210.05**	**6448.53**	**7431.07**	**6431.81**	**6236.13**	**7266.47**	**6735.94**
Fuzzy	Busy	4254.2	3895	4312.1	3875	3911.2	4243.2	4061.77

In the early runs of these systems, it was discovered that over training was occurring, and that this over training was linked to the number of nodes used in the hidden layer of the neural network. To further test this, many smaller neural networks were used for training on the system which succeeded in outperforming the standard methods. Tables 4.11 and 4.12 give the success metrics for these networks. It appears that neural networks of the size 25 and 30 hidden layer nodes perform the best depending on the metric that is more important. The 25 node network performs best for a cumulative BDM and the 30 node network performs best for the number of images better solved. Regardless it is shown that

a small network works better than larger networks. This is advantageous for implementing these systems in hardware.

Table 4.11 Fuzzy System: Comparison of Neural Network Sizes by Image Performance

NN size	10	15	20	25	30	35	40	45	50	100
Sparse GT	45	43	46	**47**	**50**	47	47	49	47	47
Busy GT	14	15	8	13	18	14	15	15	15	15

Table 4.12 Fuzzy System: Comparison of Neural Network Sizes by Cumulative BDM

NN size	10	15	20	25	30	35	40	45	50	100
Sparse GT	6567.2	6451.9	6331.6	**6210.0**	**6486.4**	6315.1	6296.3	6360.4	6332.1	6273.3
Busy GT	4612.0	4568.3	4699.0	4254.1	4232	4373.9	4431.4	4472.1	4452.4	4384.2

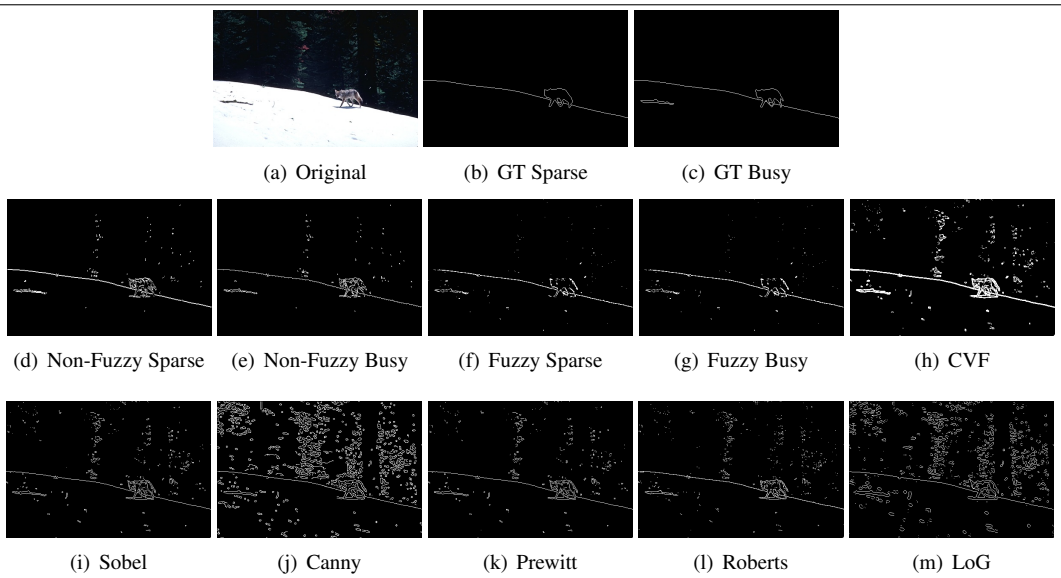
Two images were identified that performed well on all of the systems. The results of these images are given in Table 4.13. By looking at just these images there is no clear indication which system is superior. For the wolf image (167062) the Non-Fuzzy system has superior BDM but when compared in Figure 4.15 it would seem that the Fuzzy system has the better image. In contrast, for the fish image (86068) the Fuzzy system has the better BDM but the comparison of the images in Figure 4.16 shows that the non-fuzzy system is the best system for that image.

Table 4.13 Images That Do Well In All Systems With Filter and BDM for Comparison

System	Image	Filter Sparse	BDM Sparse	Filter Busy	BDM Busy
Non-Fuzzy	167062	267	89.919	200	87.465
Non-Fuzzy	86068	3	113.78	387	72.325
Fuzzy	167062	82 , 0.712 , 178	97.042	82 , 0.712 , 178	93.249
Fuzzy	86068	82 , 0.712 , 178	108.69	97 , 0.73 , 56	48.117

In order to try and increase the performance of the Non-Fuzzy system a test was performed where every image was filtered 5 times using the same CA filter selected by the system. The hypothesis is that a CA gives complex behavior through multiple time steps,

Figure 4.15 Wolf (167062)

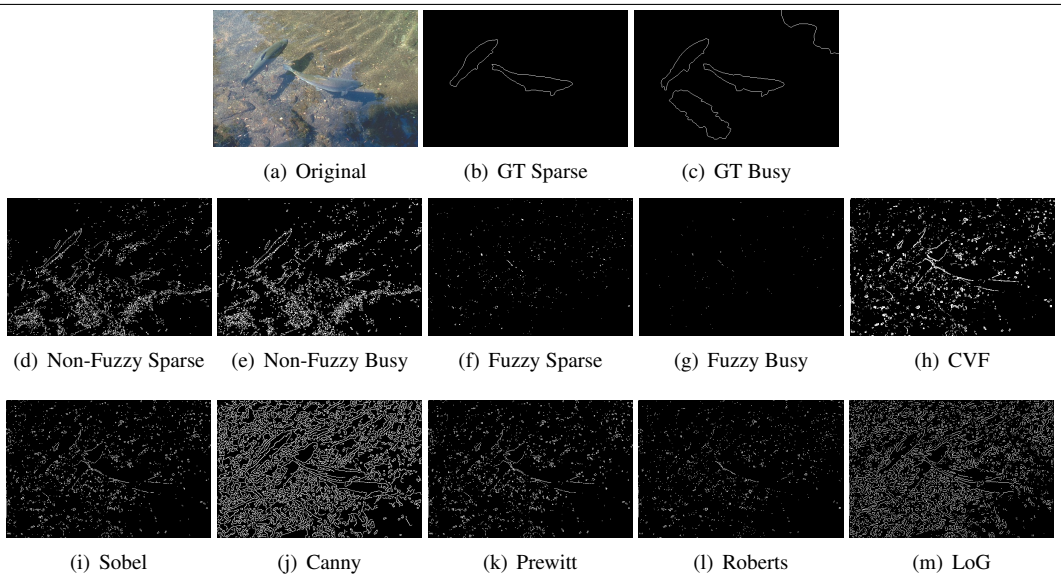


(a) Original (b) GT Sparse (c) GT Busy

(d) Non-Fuzzy Sparse (e) Non-Fuzzy Busy (f) Fuzzy Sparse (g) Fuzzy Busy (h) CVF

(i) Sobel (j) Canny (k) Prewitt (l) Roberts (m) LoG

Figure 4.16 Fish (86068)



(a) Original (b) GT Sparse (c) GT Busy

(d) Non-Fuzzy Sparse (e) Non-Fuzzy Busy (f) Fuzzy Sparse (g) Fuzzy Busy (h) CVF

(i) Sobel (j) Canny (k) Prewitt (l) Roberts (m) LoG

thus by applying the CA filter multiple times to the image a superior *edge image* will be acquired. Table 4.14 shows that this was not the case. When the filter was applied five times little change was observed over a single application. Further, the observed change was not always an improvement. Many times noise was introduced into the *edge image*. When investigating the subjective difference between the singly and multiply filtered *edge images* no appreciable difference was found in the images from the BSD, as such no example images are given.

Table 4.14 Busy Non-Fuzzy *Edge Image* BDM Comparison for *Edge Images* After a Single Application of the CA Filter and Five Applications

Image	CA	Test-Filter Run One Time	Test-Filter Run Five Times	Sobel	Canny	Prewitt	Roberts	LoG
101087	15	12.952	13.738	24.815	14.439	24.813	25.01	25.659
103070	390	25.134	26.304	37.171	26.547	37.313	36.032	30.11
123074	259	72.989	73.557	74.487	75.029	74.317	74.682	73.999
14037	137	31.702	31.082	36.377	40.459	36.53	38.804	40.999
163085	236	38.532	39.881	40.131	47.367	40.146	39.851	43
167062	200	87.465	89.882	111.42	122.08	110.95	113.71	121.02
220075	39	27.249	28.619	29.767	30.203	29.695	29.181	34.891
236037	7	23.928	24.442	25.302	24.196	24.748	36.924	23.366
253055	324	25.049	25.511	67.904	97.697	67.588	55.118	95.64
260058	422	86.414	87.59	101.4	103.27	101.39	101.06	103.19
3096	364	21.939	21.442	64.911	45.104	64.922	64.309	40.132

Appendix C gives tables for all successful images for both systems, the filters used for both systems, as well as some of the images solved by the systems for further comparison. It should be noted from these tables that not all of the filters in the filter bank were used. This could be because no scenes were present in the test set which corresponds to the unused filters, or the classification of the system could be flawed. A larger training and test set would be needed to investigate this. It is also shown in this appendix that there is no

unifying theme as to which images the systems work best on.

4.2 Test On Image Database Without Ground Truth

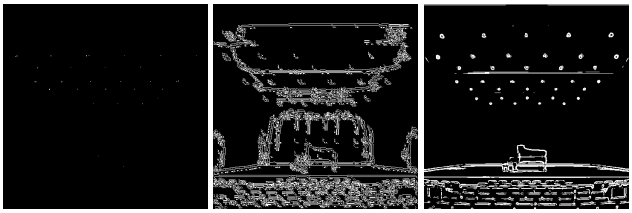
Only one of the prototype systems was shown to succeed in outperforming the standard methods, the Fuzzy system run with a 25 node hidden layer neural network. This system and its Non-Fuzzy counterpart were run against a number of images from the MIT SUN image database [37]. The Images tested were taken from the following scene types: abbey, airport, bathroom, beach, bedroom, broad leaf, candy store, house, indoor procenium, kitchen, library, and river scene. Sample result image are given in Appendix D in correspondingly labeled sections.

In general the Fuzzy system performed well on all of the outdoor scenes, the scenes that were a close up of an individual, and many scenes that were of large structures. In terms of the SUN database this would be the abbey, airport, broadleaf, indoor procenium and river scenes given by Sections D.1, D.2,D.6, D.9, D.14. The system did not perform well on images where the color pallet was fairly uniform such as the bathroom images in section D.3, or where there was a large amount of lighting contrast such as the beach images in Section D.4. Essentially, the system worked well for the images whose scenes it recognized, however when a new untrained scene was encountered often a poor filter was often chosen. The system was trained to function on the fewest edge pixel ground truths. As such, it will attempt to find the edges while accruing a small number of false positives. Hence, the system will air on missing true edges rather than chancing false positives. This tendency lead to blank *edge images* being obtained. These blank images are a result of the chosen fuzzy threshold being set too high. The improper filter was likely chosen because

of features in the image which were not encountered in the training set. The training set has only images from the BSD which all have good lighting and framing. The scenes in the SUN database are much more varied and are often images of very low quality. The difference is very well pronounce in the the stage images in Figures 4.17 and 4.18. The bright stage given by Figure 4.17 has good lighting and an edge detector which works well was selected for it. In contrast a similar stage image with poor lighting is given by Figure 4.18 which has an edge filter which performs poorly selected for it. The remedy to this issue was hypothesised to be to have a more comprehensive training set which included images with poor lighting so that the system will have filters to account for these situations.

Figure 4.17 Bright Stage

(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System

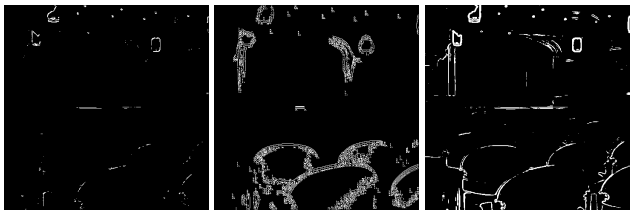


(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

Figure 4.18 Dim Stage

(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



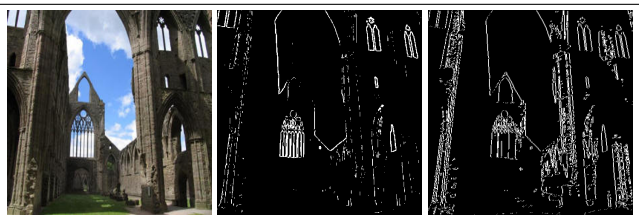
(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

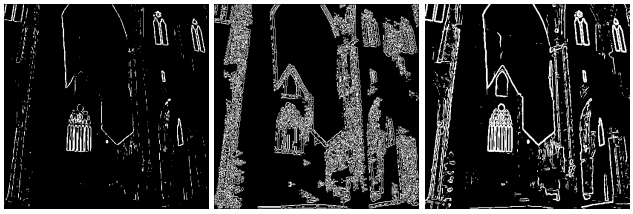
To test this hypothesis several of the images which the Fuzzy system returned a blank image were selected. For each of the selected images the best standard method *edge image* was selected as a ground truth. These new image ground truth pairs were added to the training set and a new set of filters were trained using the Fuzzy system. These filters were then used to train a new association network. This network was then applied to the images from the SUN database. This resulted in fewer blank *edge images* overall. Figure 4.18, of a cathedral, shows that the new system trained with more ground truths gives a better *edge image* than the initial system which was trained without an image with poor lighting. However, there were still misclassifications. Figure 4.17, of a bookshelf, shows that for the initial system a good filter was chosen but in the test where more images were added to the

training set a poor filter was chosen. A good filter exists for the image but it was not chosen in the second system. Though the added training images improved the systems ability to find a proper filter, the increase in filters caused more misclassificaiton.

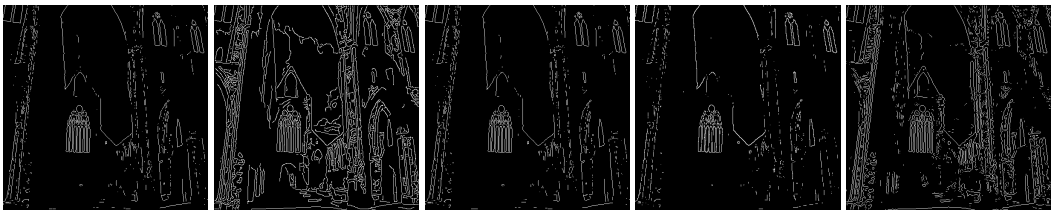
Figure 4.19 cathedral (adsyiurcswacjxze)



(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

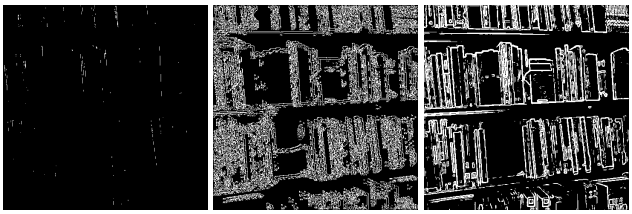


(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

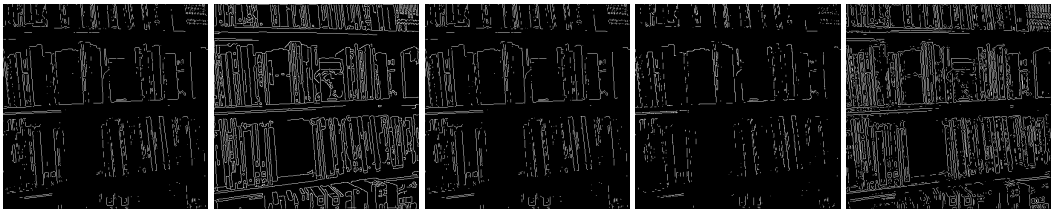
The Non-Fuzzy system was also considered. Due to the Non-Fuzzy systems poor BDM performance, it was thought that it would perform worse than the Fuzzy system in this subjective test. This was not the case. In the instances where the Fuzzy system did have an *edge image*, the Fuzzy system's *edge image* was superior to the Non-Fuzzy System's *edge image* as well as the other standard methods and the CVF. A good example would be Figure

Figure 4.20 bookshelf (ajqdbufkmatgfhkx)

(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF



(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

4.19. However, when the Fuzzy system does not produce a viable *edge image* the Non-Fuzzy system always does, and it is typically superior to the image given by the standard methods and at times even superior to CVF. This is impressive because of the additional information that the CVF is using to process the image. The decrease in clarity of results is acceptable because of the smaller amount of processing done by the filter selected by the system than by the CVF. Figure 4.20 gives a good example of this behavior, the Non-Fuzzy clearly gives a better edge image than the standard methods and the comparison to the CVF is debatable. In order to improve the Non-Fuzzy systems results the CA filter was run multiple times on the same image. This was found to only increase the width and noise of the *edge image*. The Non-Fuzzy system does not always have better *edge images* than

the Fuzzy system, however, it is more reliable and gives surprisingly good *edge images* in most cases.

Chapter 5

Conclusions

This work has shown that the Fuzzy prototype system does support the hypothesis that it would perform better than the Sobel, Canny, Prewitt, Roberts, LoG and CVF methods. This is shown by Tables 4.9 and 4.10. Table 4.9 shows that the Fuzzy prototype system successfully solves more *edge images* better than other methods. Table 4.10 shows that the Fuzzy method has a lower cumulative BDM than all other methods. The success of the system is further seen by comparing the *edge images* given by the Fuzzy prototype system to the *edge images* given by the standard methods and CVF in Appendix C. The Fuzzy system was shown to be subjectively the superior system to the standard methods and performed well against the CVF. It is acceptable for the filter using grey scale values to be of slightly lower quality because of the smaller amount of processing required. The test on the images with no ground truth showed the the Non-Fuzzy system to be superior. This superiority was because of its reliability. The Fuzzy system still produced better *edge images*, but was not consistent, often giving blank *edge images* for scenes that the system could not categorize. For these same images, the Non-Fuzzy system still performed better than the standard methods and at times performed better than even the CVF method.

The prototype systems have shown that a neural network can be trained to create Cellular

Automata edge filters based on the input image itself. By showing this, the system also shows that many Cellular Automata are able to behave well as edge detectors for a given image. It was further shown by the Fuzzy system that the quality of the trained *edge image* produced by the Non-Fuzzy system could be improved by using a fuzzy rule. This increase in quality came at the price of generality. Regardless the network was able to reliably associate functional edge filters with input images. This means that the edge properties of an image may be linked to its scene class. Further the network is able to learn these properties and use them.

All of these findings taken together shown that using the proposed framework it is possible to train a system to assign filters to images which will perform better then all of the standard methods on average across a wide range of image types. Thus the framework is a valid alternative to the standard methods for edge detection in multi-scene applications.

Chapter 6

Future Work

This Thesis was merely the first step in the research of developing a system to generate edge detection filters specific to the scenes in which the user of the edge detection finds themselves. As stated many times, this system was designed to be modular and as such each portion of the system can and should be replaced with new modules that fit the functionality but will improve overall performance.

The most important module to refine would be the Learning Step. Section 2.1.2 gives a number of methodologies for characterizing and comparing *edge image*. The largest issues experienced by this work has been poor *edge image* characterization. The BDM was selected because it seemed to handle the largest number of characterization errors and even it was a poor tool for determining the error of an *edge image*. Thus, replacing the BDM with a yet better tool would improve the whole process. In addition, the BSD should be replaced with an *edge image* database. The segmentation database is too strict to properly train an edge detector. Also, the new database should include images of all quality. It was seen, in this work, that there are large gaps in the system's knowledge when only high quality images are used. In a true application, poor quality images will need to be processed reliably and that will only happen if that eventuality is trained for. Finally, the ground truths

in the database should have unified criteria for an edge.

Another direction that could be taken for the learning step would be to create an interactive training method. This method will ask the user for input after each round of training. The input would be an inquiry to determine if the *edge image* is acceptable. The training could then be made interactive. Each new image could give the user the opportunity to give the system feedback to learn from.

The next module to refine would be the edge detection filter creation. The CA filters could be improved by using more than just the linear CA neighborhoods, or by developing a different transition rule for the neighborhoods, or both. Another improvement to the CA filter would be to find a method to use color information so as to allow the CA to take advantage of more information and potentially create better *edge images*. Care would need to be taken to not aggravate the already present filter misclassification problem by keeping the filters general enough. In addition, other methods of edge detection should be investigated. Section 2.1.5 contains a number of edge detection methodologies which could be adapted to serve as filter generators. To better understand the adaptability of the framework a number of these methods would need to be tried.

The feature extraction module could also be revised. It was seen through experimentation that the lighting of images had a large impact on the edge filter association. Some image preprocessing could be added to diminish the effects of large amounts of lighting contrast. Alternatively, another method of feature extraction could be selected. There are a number to choose from, some of which are given in Section 2.1.8. Regardless, the misclassification error would need to be reduced.

The filter selection module could also be changed. The methodology used was the simplest topology for a neural network, a single hidden layer feed forward network. This was appealing because of its small size and potential for easy implementation in hardware for fast processing. A more complex topology may yield better results but that would need to be weighed against the possible uses and hardware implementations. It is also possible to use an entirely different selection technique.

Once a system has been settled on the next step would be to implement it in hardware and integrate it into a larger application. The application would be a mobile robot which is traveling through multiple scenes. The system could then be evaluated as to how much it improves the robot's ability to perform its tasks.

Bibliography

[1] E. W. Weisstein. Rule 250. From MathWorld–A Wolfram Web Resource. [Online]. Available: http://mathworld.wolfram.com/Rule250.html

[2] ——. Rule 222. From MathWorld–A Wolfram Web Resource. [Online]. Available: http://mathworld.wolfram.com/Rule222.html

[3] ——. Rule 30. From MathWorld–A Wolfram Web Resource. [Online]. Available: http://mathworld.wolfram.com/Rule30.html

[4] ——. Rule 110. From MathWorld–A Wolfram Web Resource. [Online]. Available: http://mathworld.wolfram.com/Rule110.html

[5] C. Siagian and L. Itti, "Rapid biologically-inspired scene classification using features shared with visual attention," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 2, pp. 300–312, 2007.

[6] Y.-T. Hsiao, C.-L. Chuang, J.-A. Jiang, and C.-C. Chien, "A contour based image segmentation algorithm using morphological edge detection," in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, Oct 2005, pp. 2962–2967 Vol. 3.

[7] M. Murshed, A. Ramirez, and O. Chae, "Statistical background modeling: An edge segment based moving object detection approach," in *Seventh IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Aug 2010, pp. 300–306.

[8] H. Jin, Y. Yoshitomo, and M. Sakauchi, "Detection of information relating to building object in news video," in *International Conference on Image Processing (ICIP)*, vol. 3, 1999, pp. 329–333 vol.3.

[9] S. Konishi, A. L. Yuille, J. M. Coughlan, and Z. Song-Chun, "Statistical edge detection: learning and evaluating edge cues," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 1, pp. 57–74, 2003.

[10] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, *Digital Image Processing using MATLAB*. Pearson, 2004, pg. 541-550.

[11] R. Medina-Carnicer, R. Munoz-Salinas, E. Yeguas-Bolivar, and L. Diaz-Mas, "A novel method to look for the hysteresis thresholds for the canny edge detector," *Pattern Recognition*, vol. 44, no. 6, pp. 1201–1211, 2011.

[12] S. Uguz, U. Sahin, and F. Sahin, "Uniform cellular automata linear rules for edge detection," in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2013, Conference Proceedings, pp. 2945–2950.

[13] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, Nov 1986.

[14] S. C. Nercessian, S. S. Agaian, and K. A. Panetta, "A new reference-based measure for objective edge map evaluation," in *Mobile Multimedia/Image Processing, Security, and Applications (SPIE)*, vol. 7351, 2014, Conference Proceedings, pp. 73 510J–73 510J–12.

[15] A. Baddeley, "An error metric for binary images," in *Robust Computer Vision*, W. Forstner and S. Ruwiedel, Eds., march 1992, pp. 59–78.

[16] F. Wenlong, M. Johnston, and Z. Mengjie, "Figure of merit based fitness functions in genetic programming for edge detection," in *9th International Conference on Simulated Evolution and Learning. (SEAL)*. Springer Verlag, 2012, Conference Proceedings, pp. 22–31.

[17] A. Pinho and s. Almeida, Lua, "Edge detection filters based on artificial neural networks," in *Image Analysis and Processing*, ser. Lecture Notes in Computer Science, C. Braccini, L. DeFloriani, and G. Vernazza, Eds. Springer Berlin Heidelberg, 1995, vol. 974, pp. 159–164.

[18] S. Uguz, U. Sahin, and F. Sahin, "Edge detection with fuzzy cellular automata transition function optimized by pso," *Computers & Electrical Engineering*, 2015.

[19] I. Kokkinos, *Boundary Detection Using F-Measure-, Filter- and Feature- (F3) Boost*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, vol. 6312, book section 47, pp. 650–663.

[20] F. Wenlong, M. Johnston, and Z. Mengjie, "Unsupervised learning for edge detection using genetic programming," in *IEEE Congress on Evolutionary Computation (CEC)*, 2014, Conference Proceedings, pp. 117–124.

[21] Y. Yitzhaky and E. Peli, "A method for objective edge detection evaluation and detector parameter selection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, pp. 1027–1033, 2003.

[22] R. Koren and Y. Yitzhaky, "Automatic selection of edge detector parameters based on spatial and statistical measures," *Computer Vision and Image Understanding*, vol. 102, no. 2, pp. 204–213, 2006.

[23] X. Xia, B. Li, and W. Chengdong, "Automatic parameters selection method of edge detector in the unstructured environment," in *IEEE International Conference on Robotics and Biomimetics*, 2008, Conference Proceedings, pp. 1740–1743.

[24] M. Bennamoun, B. Boashash, and J. Koo, "Optimal parameters for edge detection," in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, 1995, Conference Proceedings, pp. 1482–1488 vol.2.

[25] C. Yuchou, L. Dah-Jye, H. Yi, and J. Archibald, "Edge detection from global and local views using an ensemble of multiple edge detectors," in *4th International Symposium Advances in Visual Computing (ISVC)*, vol. pt.2. Springer-Verlag, 2008, Conference Proceedings, pp. 934–41.

[26] L. Mason and J. Baxter, "Boosting Algorithms as Gradient Descent," in *Neural Information Processing Systems*, 2000.

[27] C. Lopez-Molina, B. De Baets, and H. Bustince, "Generating fuzzy edge images from gradient magnitudes," *Computer Vision and Image Understanding*, vol. 115, no. 11, pp. 1571–1580, 2011.

[28] H.-Y. Wang, H.-D. Li, X.-Q. Ye, and W.-K. Gu, "Training a neural network for moment based image edge detection," *Journal of Zhejiang University (Science)*, vol. 1, no. 4, pp. 398–401, 2000.

[29] K. S. Komati, E. O. T. Salles, and M. Sarcinelli-Filho, "A bio-inspired system for boundary detection in color natural scenes," in *12th International Conference on Computational Science and Its Applications (ICCSA)*, vol. pt.2. Springer Verlag, 2012, Conference Proceedings, pp. 739–52.

[30] D. Ziou and A. Koukam, "Knowledge-based assistant for the selection of edge detectors," *Pattern Recognition*, vol. 31, no. 5, pp. 587–596, 1998.

[31] C. D. Thomas, "Evolution of cellular automata for image processing," Thesis, University of Birmingham (UK), 2000.

[32] Q. K. K. A. Fasel, "Investigations of cellular automata linear rules for edge detection," *IJCNIS*, vol. 4, no. 3, pp. 47–53, 2012.

[33] K. Mirzaei, H. Motameni, and R. Enayatifar, "New method for edge detection and de noising via fuzzy cellular automata," *International Journal of Physical Sciences*, vol. 6, no. 13, pp. 3175–3180, 2011.

[34] S. Sinaie, A. Ghanizadeh, E. M. Majd, and S. M. Shamsuddin, "A hybrid edge detection method based on fuzzy set theory and cellular learning automata," in *International Conference on Computational Science and Its Applications (ICCSA)*, 2009, Conference Proceedings, pp. 208–214.

[35] E. Alypaydin, *Introduction to Machine Learning*, T. Dietterich, Ed. MIT Press, 2010, pg.113-120, ch 11.

[36] S. A. Dianat and E. Saber, *Advanced Linear Algebra for Engineeris with Matlab*. CRC Press, 2009.

[37] X. Jianxiong, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, Conference Proceedings, pp. 3485–3492.

[38] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, Conference Proceedings, pp. 413–420.

[39] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2006, Conference Proceedings, pp. 2169–2178.

[40] H. Madokoro, Y. Utsumi, and K. Sato, "Scene classification using unsupervised neural networks for mobile robot vision," in *SICE Annual Conference (SICE), 2012 Proceedings of*, 2012, Conference Proceedings, pp. 1568–1573.

[41] C. Pavlopoulou and S. X. Yu, "Indoor-outdoor classification with human accuracies: Image or edge gist?" in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2010, Conference Proceedings, pp. 41–47.

[42] M. Xianglin and W. Zheng-Zhi, "Rapid scene categorization using novel gist model," in *International Conference on Information Engineering and Computer Science (ICIECS)*, 2010, Conference Proceedings, pp. 1–4.

[43] C. Siagian and L. Itti, "Gist: A mobile robotics application of context-based vision in outdoor environment," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, (CVPR Workshops)*, 2005, Conference Proceedings, pp. 88–88.

[44] S. Wolfram, "A new kind of science," 2002.

[45] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *IEEE International Conference on Neural Networks*, vol. 4, Nov 1995, pp. 1942–1948 vol.4.

[46] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 530–549, 2004.

[47] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2010, pp. 3485–3492.

[48] H.-C. Lee and D. Cok, "Detecting boundaries in a vector field," *Signal Processing, IEEE Transactions on*, vol. 39, no. 5, pp. 1181–1194, May 1991.

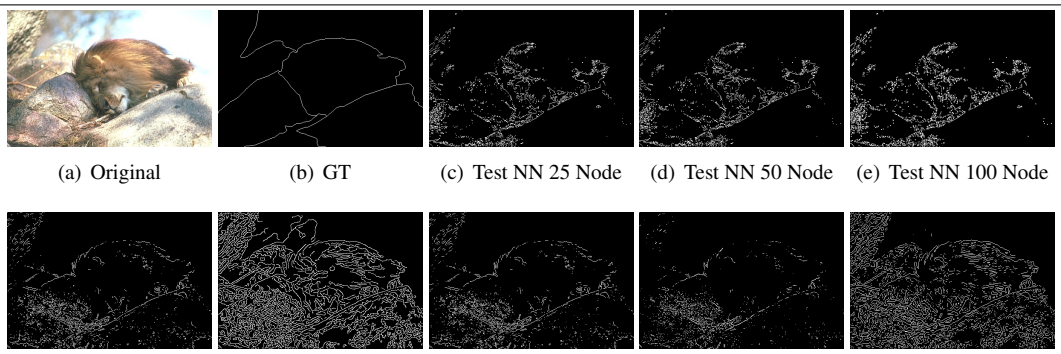
Appendix A

Successful *Edge Images* from Initial Test Run

Table A.1 Individual BDM Error: Filter Selected by System Trained and Tested Against Ground Truth Index 1

Image Name	Filter # NN 25	BDM NN 25	Filter # NN 50	BDM NN 50	Filter # NN 100	BDM NN 100
105025	7	39.212	7	39.212	15	39.224
14037	7	34.092	7	34.092	7	34.092
157055	94	15.677	86	15.672	94	15.677
167062	246	88.315	162	87.848	246	88.315
220075	416	34.169	419	34.2	433	34.256
227092	207	38.044	206	38.044	78	38.027
253055	23	25.365	275	25.411	279	25.441
260058	124	86.514	124	86.514	60	86.714
296007	44	16.688	61	16.668	60	16.668
296059	47	27.069	39	27.039	38	27.039
306005	7	72.626	7	72.626	7	72.626
41033	23	42.117	151	42.236	151	42.236
42012	464	26.862	464	26.862	464	26.862
43074	308	52.285	60	52.175	60	52.175
62096	47	66.029	324	66.134	302	66.127
69040	44	44.103	44	44.103	44	44.103
85048	39	31.143	39	31.143	55	31.194
86068	83	73.313	323	73.194	451	73.164
87046	414	17.831	158	17.787	414	17.831
119082	409	34.378	473	39.337	473	39.337

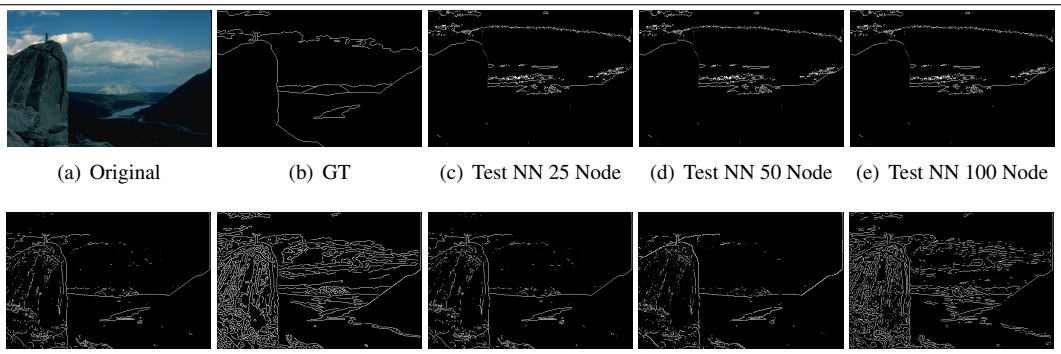
Figure A.1 105025



(a) Original (b) GT (c) Test NN 25 Node (d) Test NN 50 Node (e) Test NN 100 Node

(f) Sobel (g) Canny (h) Prewitt (i) Roberts (j) LoG

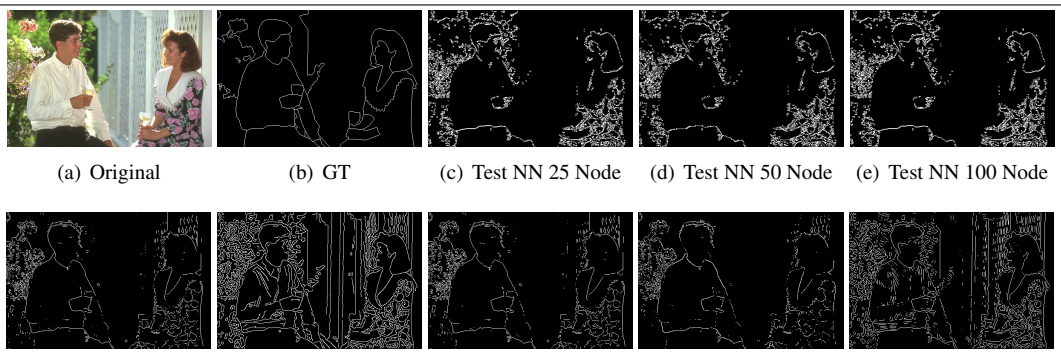
Figure A.2 14037



(a) Original (b) GT (c) Test NN 25 Node (d) Test NN 50 Node (e) Test NN 100 Node

(f) Sobel (g) Canny (h) Prewitt (i) Roberts (j) LoG

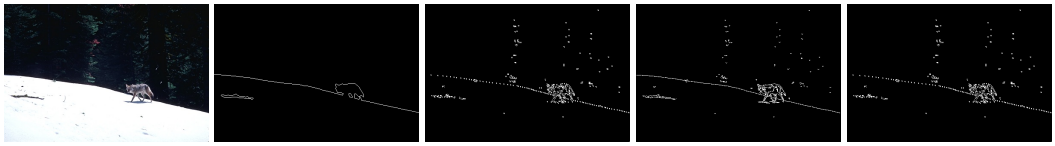
Figure A.3 157055



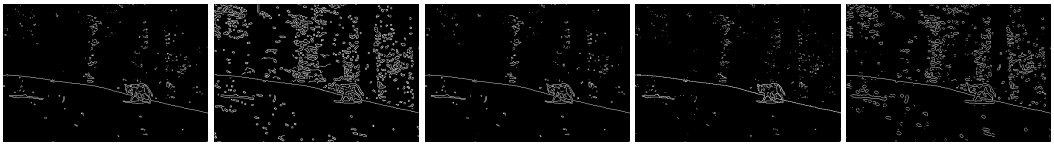
(a) Original (b) GT (c) Test NN 25 Node (d) Test NN 50 Node (e) Test NN 100 Node

(f) Sobel (g) Canny (h) Prewitt (i) Roberts (j) LoG

Figure A.4 167062

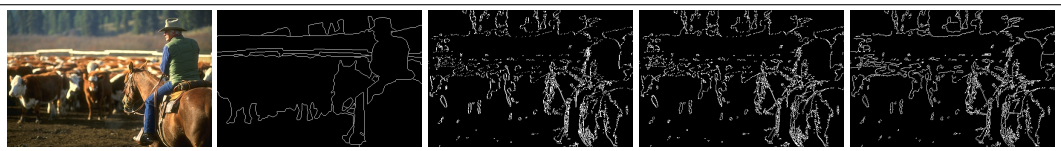


(a) Original (b) GT (c) Test NN 25 Node (d) Test NN 50 Node (e) Test NN 100 Node



(f) Sobel (g) Canny (h) Prewitt (i) Roberts (j) LoG

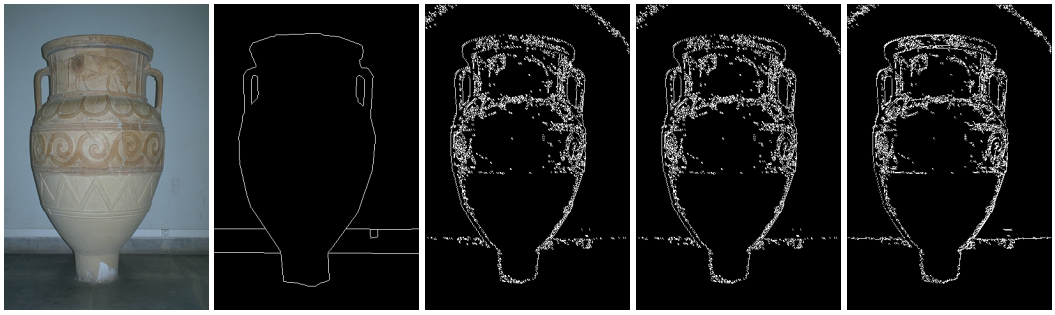
Figure A.5 220075



(a) Original (b) GT (c) Test NN 25 Node (d) Test NN 50 Node (e) Test NN 100 Node

(f) Sobel (g) Canny (h) Prewitt (i) Roberts (j) LoG

Figure A.6 227092



(a) Original (b) GT (c) Test NN 25 Node (d) Test NN 50 Node (e) Test NN 100 Node

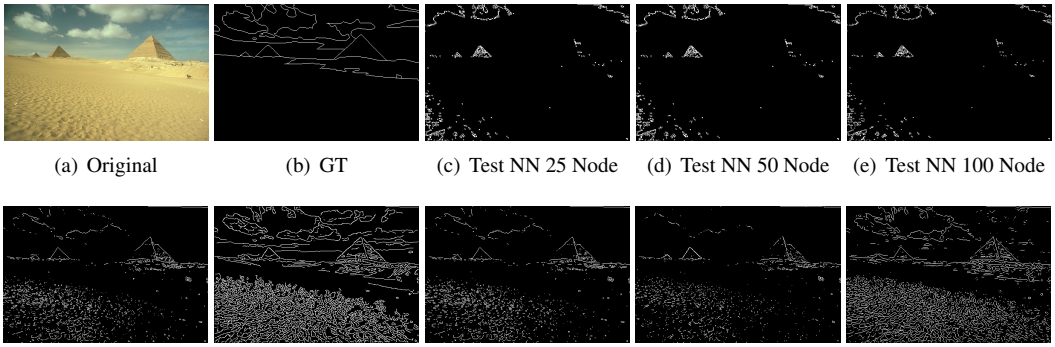
(f) Sobel (g) Canny (h) Prewitt (i) Roberts (j) LoG

Figure A.7 253055

(a) Original (b) GT (c) Test NN 25 Node (d) Test NN 50 Node (e) Test NN 100 Node

(f) Sobel (g) Canny (h) Prewitt (i) Roberts (j) LoG

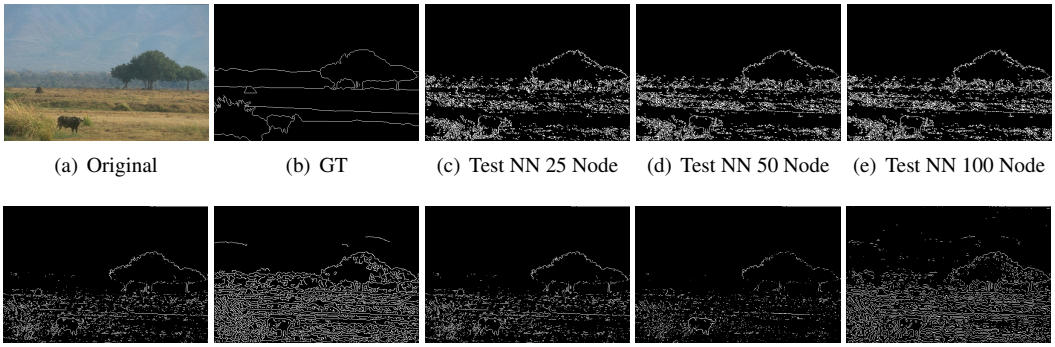
Figure A.8 260058



(a) Original (b) GT (c) Test NN 25 Node (d) Test NN 50 Node (e) Test NN 100 Node

(f) Sobel (g) Canny (h) Prewitt (i) Roberts (j) LoG

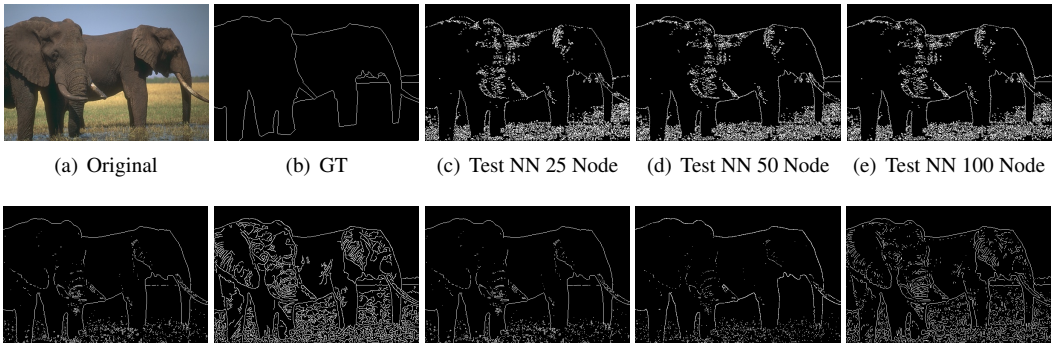
Figure A.9 296007



(a) Original (b) GT (c) Test NN 25 Node (d) Test NN 50 Node (e) Test NN 100 Node

(f) Sobel (g) Canny (h) Prewitt (i) Roberts (j) LoG

Figure A.10 296059



(a) Original (b) GT (c) Test NN 25 Node (d) Test NN 50 Node (e) Test NN 100 Node

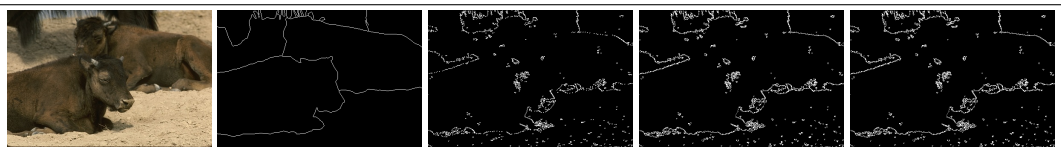
(f) Sobel (g) Canny (h) Prewitt (i) Roberts (j) LoG

Figure A.11 306005

(a) Original (b) GT (c) Test NN 25 Node (d) Test NN 50 Node (e) Test NN 100 Node

(f) Sobel (g) Canny (h) Prewitt (i) Roberts (j) LoG

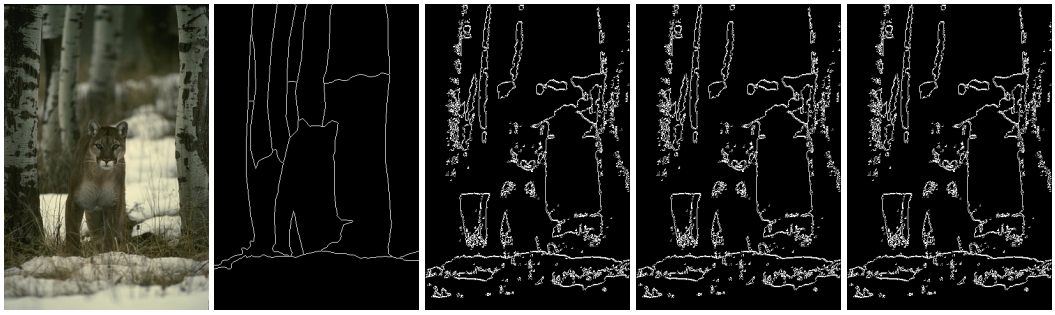
Figure A.12 41033



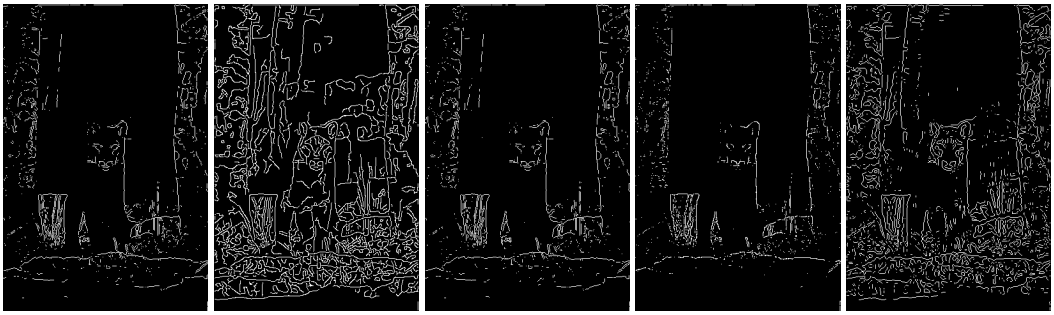
(a) Original (b) GT (c) Test NN 25 Node (d) Test NN 50 Node (e) Test NN 100 Node

(f) Sobel (g) Canny (h) Prewitt (i) Roberts (j) LoG

Figure A.13 42012

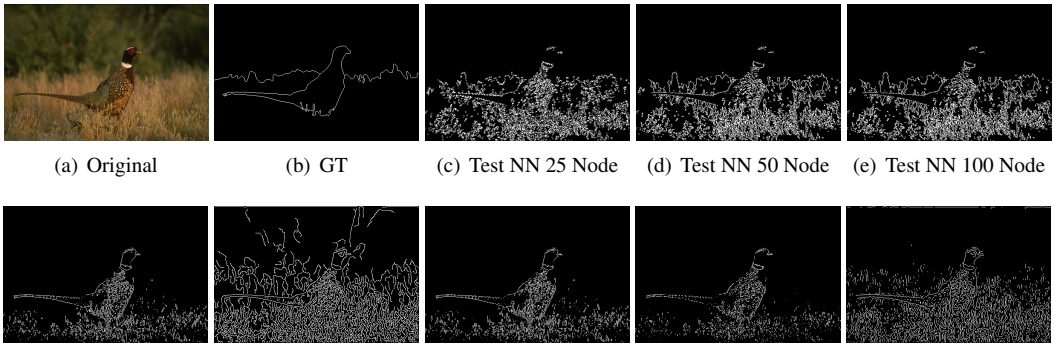


(a) Original (b) GT (c) Test NN 25 Node (d) Test NN 50 Node (e) Test NN 100 Node



(f) Sobel (g) Canny (h) Prewitt (i) Roberts (j) LoG

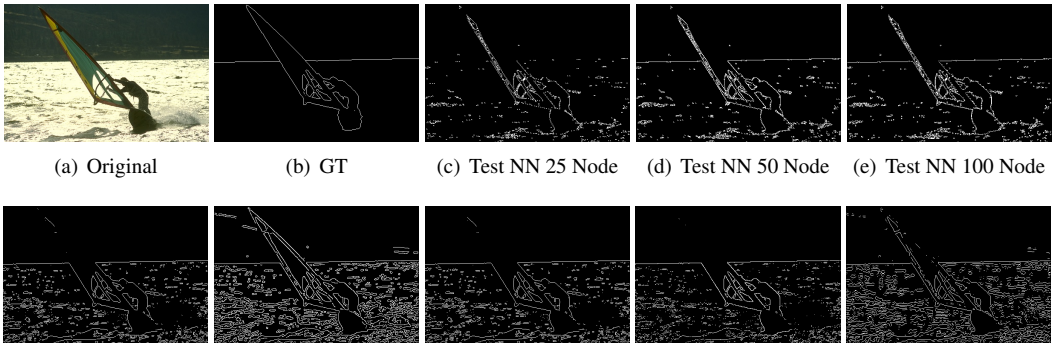
Figure A.14 43074



(a) Original (b) GT (c) Test NN 25 Node (d) Test NN 50 Node (e) Test NN 100 Node

(f) Sobel (g) Canny (h) Prewitt (i) Roberts (j) LoG

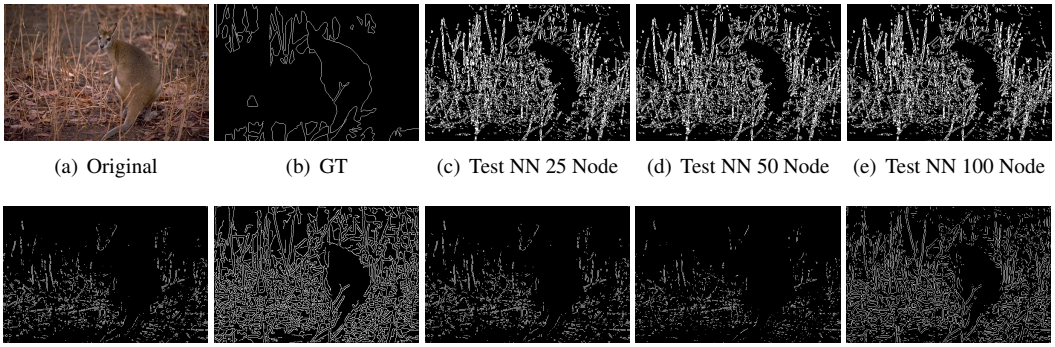
Figure A.15 62096



(a) Original (b) GT (c) Test NN 25 Node (d) Test NN 50 Node (e) Test NN 100 Node

(f) Sobel (g) Canny (h) Prewitt (i) Roberts (j) LoG

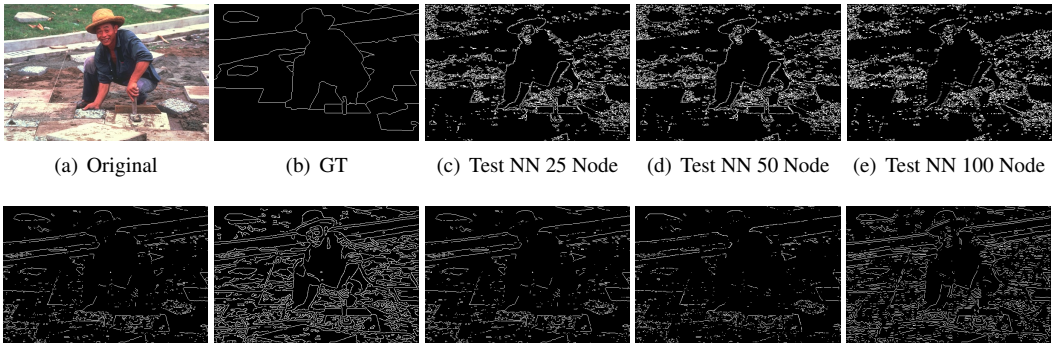
Figure A.16 69040



(a) Original (b) GT (c) Test NN 25 Node (d) Test NN 50 Node (e) Test NN 100 Node

(f) Sobel (g) Canny (h) Prewitt (i) Roberts (j) LoG

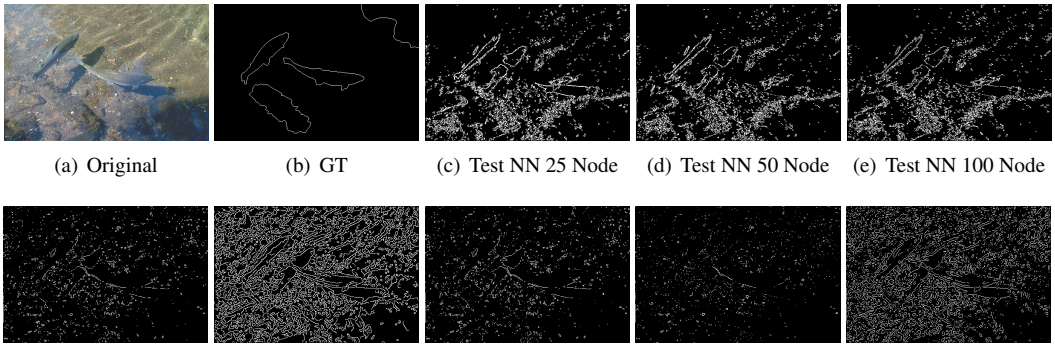
Figure A.17 85048



(a) Original (b) GT (c) Test NN 25 Node (d) Test NN 50 Node (e) Test NN 100 Node

(f) Sobel (g) Canny (h) Prewitt (i) Roberts (j) LoG

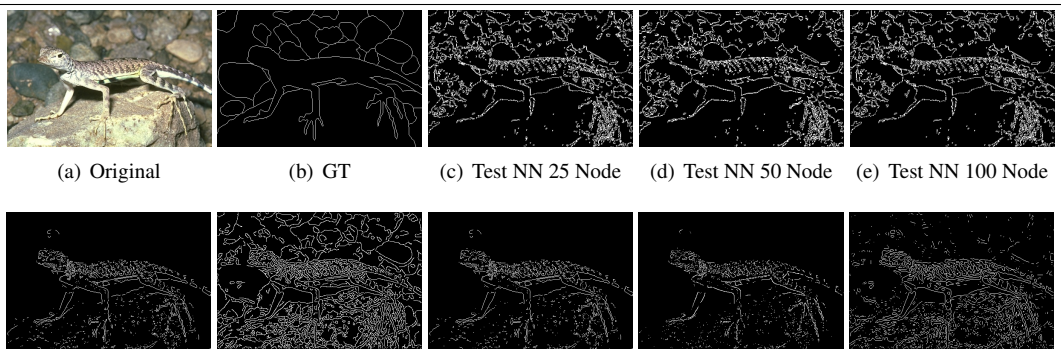
Figure A.18 86068



(a) Original (b) GT (c) Test NN 25 Node (d) Test NN 50 Node (e) Test NN 100 Node

(f) Sobel (g) Canny (h) Prewitt (i) Roberts (j) LoG

Figure A.19 87046



(a) Original (b) GT (c) Test NN 25 Node (d) Test NN 50 Node (e) Test NN 100 Node

(f) Sobel (g) Canny (h) Prewitt (i) Roberts (j) LoG

Appendix B

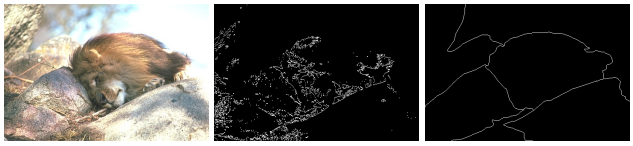
Successful images from Second Run

B.1 Non-Fuzzy Images

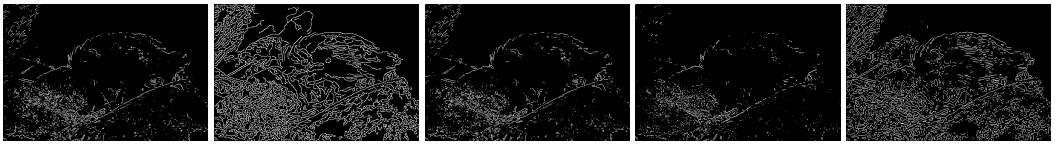
Table B.1 BDM and Filters of Successful *edge images* From Non-Fuzzy System Using 25 Hidden Nodes With Comparison BDM from Standard Methods

Image	CA	Test	Sobel	Canny	Prewitt	Robert	LoG
105025	7	39.212	42.553	47.737	42.497	40.506	46.546
119082	409	34.378	36.571	40.946	36.642	36.505	40.346
14037	7	34.092	52.716	56	52.789	54.242	56.268
157055	94	15.677	20.219	28.345	20.695	19.482	27.397
167062	246	88.315	111.88	122.54	111.42	114.17	121.49
220075	416	34.169	37.024	37.593	36.945	36.126	42.104
227092	207	38.044	38.227	43.719	38.066	38.471	42.527
253055	23	25.365	68.003	97.903	67.643	55.188	95.859
260058	124	86.514	101.4	103.27	101.39	101.06	103.19
296007	44	16.688	30.034	33.809	30.325	31.245	61.519
296059	47	27.069	28.895	35.789	28.837	27.686	35.178
41033	23	42.117	44.81	52.721	44.632	45.259	52.152
42012	464	26.862	33.773	32.947	33.422	33.35	33.603
43074	308	52.285	63.065	84.254	63.191	62.988	77.459
62096	47	66.029	69.718	72.226	69.895	70.265	71.709
85048	39	31.143	33.115	35.878	33.044	32.431	35.567
87046	414	17.831	54.445	24.189	54.436	55.086	23.655

Figure B.1 Image 105025



(a) Original (b) Test (c) GT



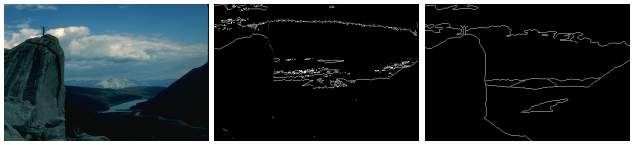
(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

Figure B.2 Image 119082

(a) Original (b) Test (c) GT

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

Figure B.3 Image 14037



(a) Original (b) Test (c) GT

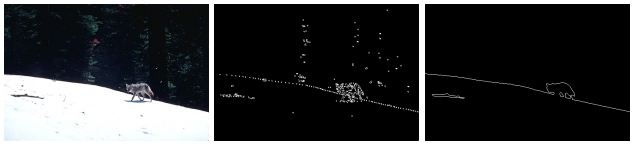
(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

Figure B.4 Image 157055

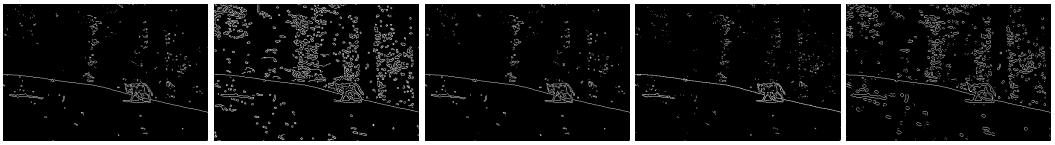
(a) Original (b) Test (c) GT

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

Figure B.5 Image 167062

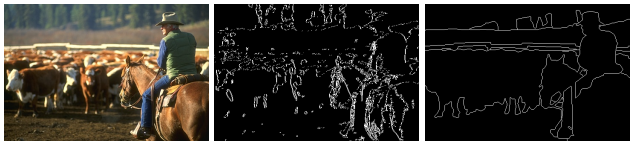


(a) Original (b) Test (c) GT



(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

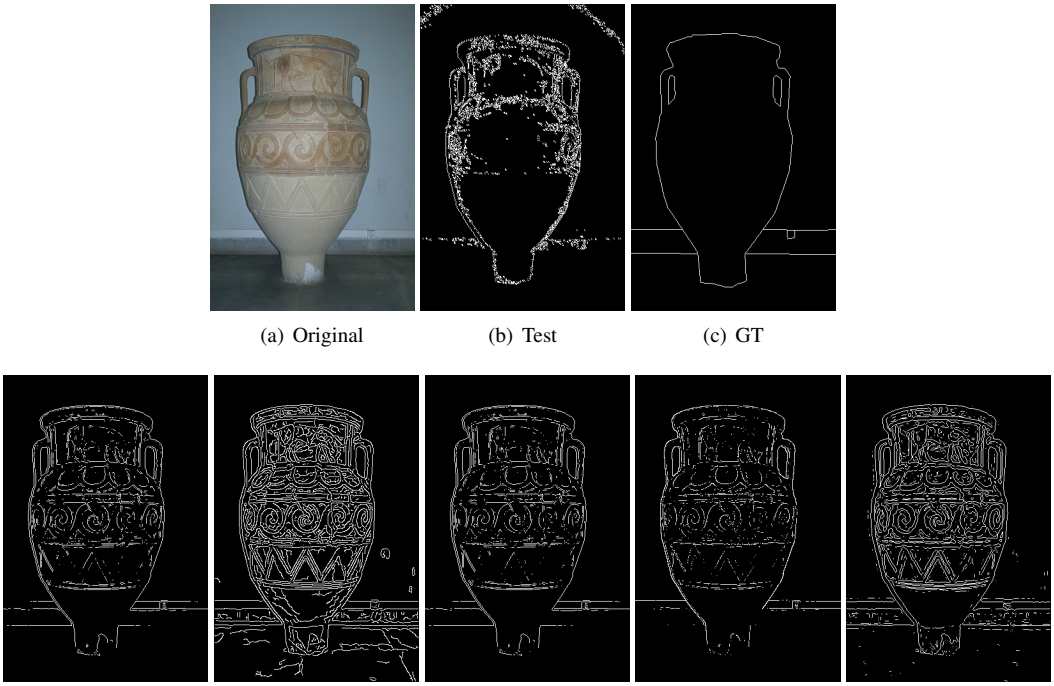
Figure B.6 Image 220075



(a) Original (b) Test (c) GT

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

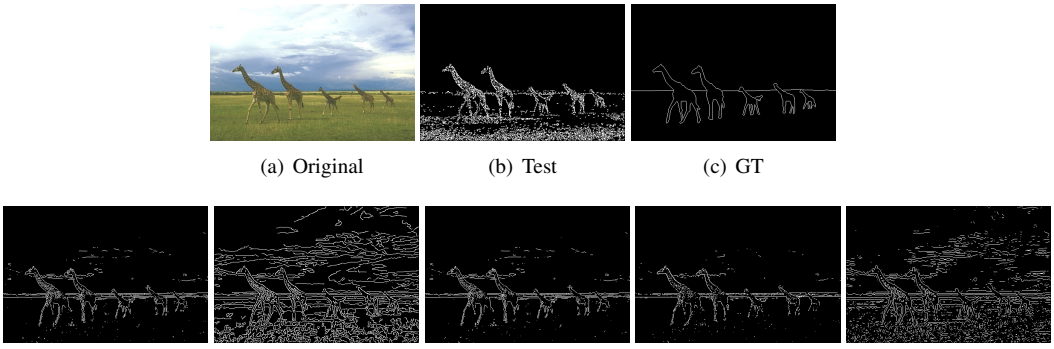
Figure B.7 Image 227092



(a) Original (b) Test (c) GT

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

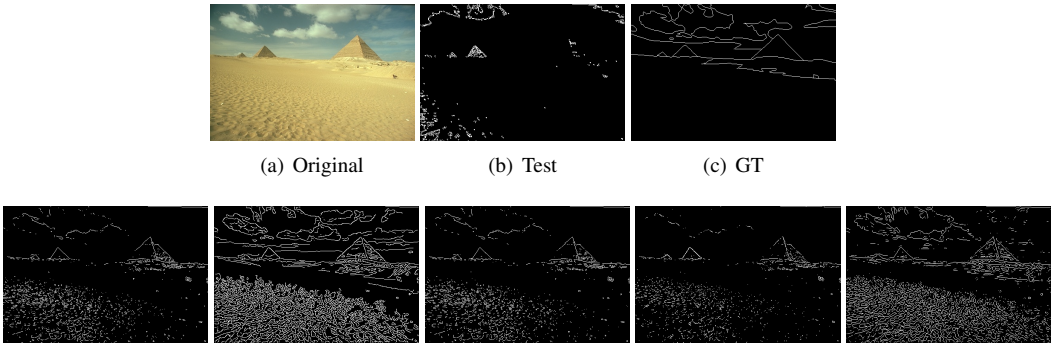
Figure B.8 Image 253055



(a) Original (b) Test (c) GT

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

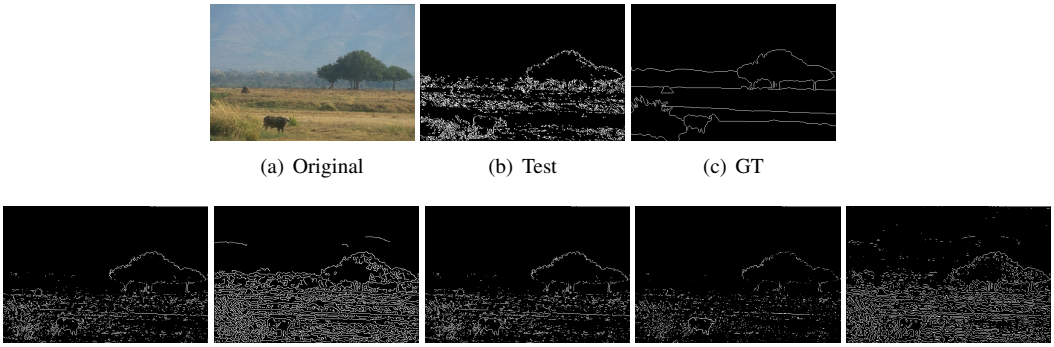
Figure B.9 Image 260058



(a) Original (b) Test (c) GT

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

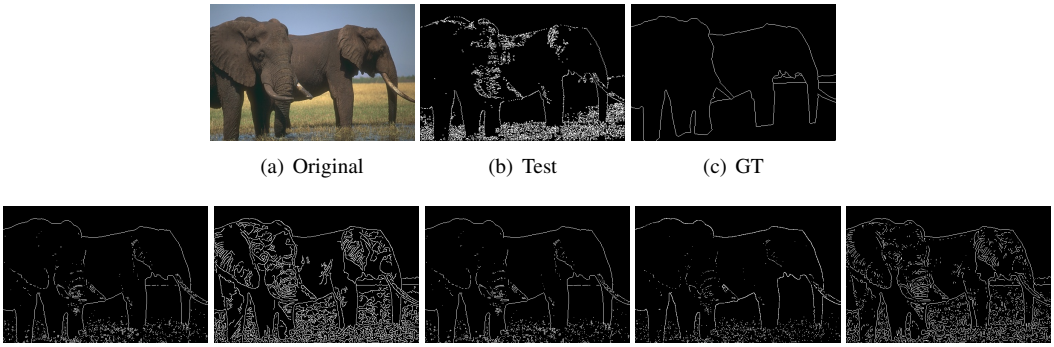
Figure B.10 Image 296007



(a) Original (b) Test (c) GT

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

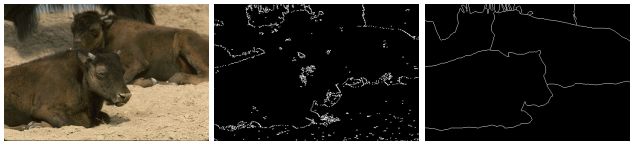
Figure B.11 Image 296059



(a) Original (b) Test (c) GT

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

Figure B.12 Image 41033

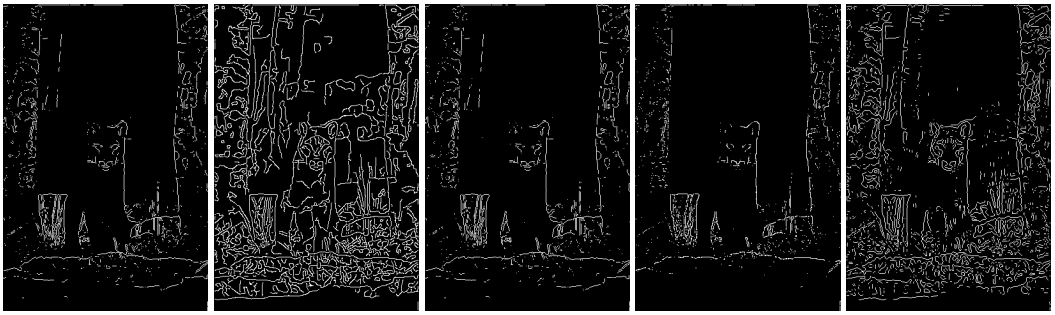


(a) Original (b) Test (c) GT

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

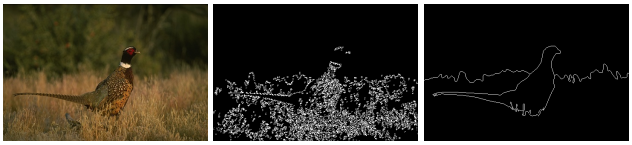
Figure B.13 Image 42012

(a) Original (b) Test (c) GT



(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

Figure B.14 Image 43074



(a) Original (b) Test (c) GT

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

Figure B.15 Image 62096

(a) Original (b) Test (c) GT

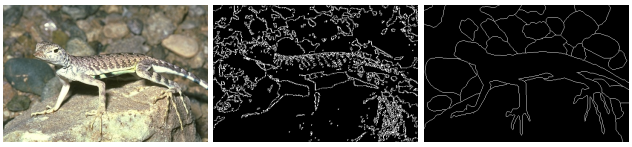
(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

Figure B.16 Image 85048

(a) Original (b) Test (c) GT

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

Figure B.17 Image 87046



(a) Original (b) Test (c) GT

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

B.2 Fuzzy Images

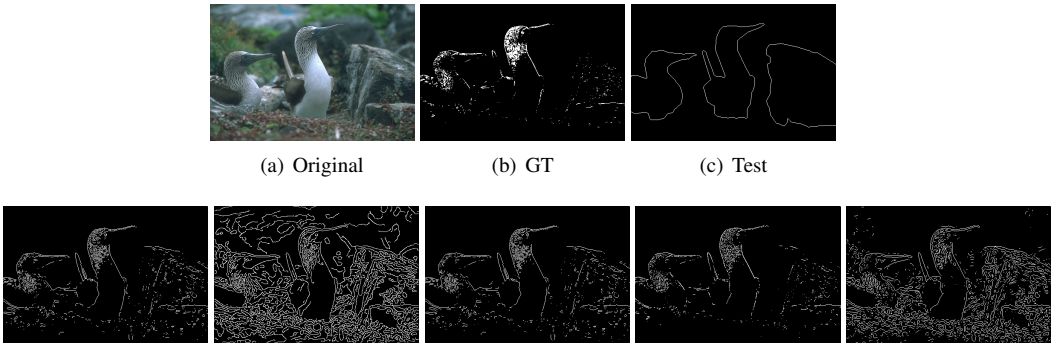
Table B.2 Fuzzy System Filters Selected by Training Processes

OffSet	Threshold	CA
40	0.23182	78
124	0.67	203
92	0.6	326
73	0.43	168
101	0.34	121
134	0.3	18
82	0.712	178
86	0.87	245
39	0.4	45
35	0.71	45
112	0.54	410
68	0.47	386
71	0.19	355
45	0.47906	283
97	0.73	56
60	0.3	23
51	0.23	123

Table B.3 BDM and Filters of Successful *edge images* from Fuzzy System using 25 Hidden Nodes with Comparison BDM from Standard Methods

Image	OffSet	Threshold	CA	Test	Sobel	Canny	Prewitt	Robert	LoG
103070	35	0.71	45	25.5	28.506	49.591	28.648	26.817	41.592
119082	82	0.712	178	31.485	36.571	40.946	36.642	36.505	40.346
12084	82	0.712	178	44.202	74.036	82.866	73.851	70.372	82.462
130026	82	0.712	178	75.696	87.128	93.264	85.671	80.547	93.265
145086	82	0.712	178	69.121	74.127	84.59	73.736	72.295	82.966
147091	82	0.712	178	28.177	32.618	52.688	32.119	31.824	45.711
148089	82	0.712	178	20.693	22.121	25.74	21.995	21.256	25.606
157055	35	0.71	45	18.508	20.219	28.345	20.695	19.482	27.397
16077	82	0.712	178	34.356	40.779	44.318	40.743	40.408	44.376
163085	35	0.71	45	36.241	38.875	54.346	38.878	38.051	50.659
167062	82	0.712	178	93.829	111.88	122.54	111.42	114.17	121.49
175043	82	0.712	178	100.89	125.72	134.13	126.26	106.95	133.67
189080	92	0.6	326	26.124	30.421	36.654	28.597	30.85	36.605
210088	35	0.71	45	123.6	126.95	133.86	126.23	125.43	132.33
236037	82	0.712	178	55.528	62.512	71.858	62.315	59.644	71.099
253027	82	0.712	178	14.692	16.444	46.649	16.565	15.409	24.74
253055	82	0.712	178	17.777	68.003	97.903	67.643	55.188	95.859
304034	82	0.712	178	71.184	84.506	92.658	84.846	74.043	92.198
306005	82	0.712	178	61.829	72.744	76.468	72.571	72.217	79.272
3096	35	0.71	45	2.1641	45.142	115.97	45.142	45.445	107.91
33039	97	0.73	56	108.77	110.42	112.43	110.33	109.31	112.11
38092	82	0.712	178	29.054	31.953	39.994	31.745	31.254	34.91
42012	92	0.6	326	32.117	33.773	32.947	33.422	33.35	33.603
66053	82	0.712	178	37.687	64.588	69.003	64.794	44.601	68.859
78004	134	0.3	18	33.67	35.008	39.381	35.942	34.67	39.192
8023	82	0.712	178	28.187	54.605	95.056	54.726	38.365	92.147
86000	82	0.712	178	36.846	37.428	40.995	37.46	37.112	46.05
86016	82	0.712	178	40.131	40.277	40.952	40.312	42.212	40.735
86068	82	0.712	178	67.512	74.582	79.424	73.9	70.819	79.273
89072	82	0.712	178	29.247	35.386	36.84	35.344	34.949	36.983
97033	82	0.712	178	106.84	109.71	113.81	109.6	108.43	113.31

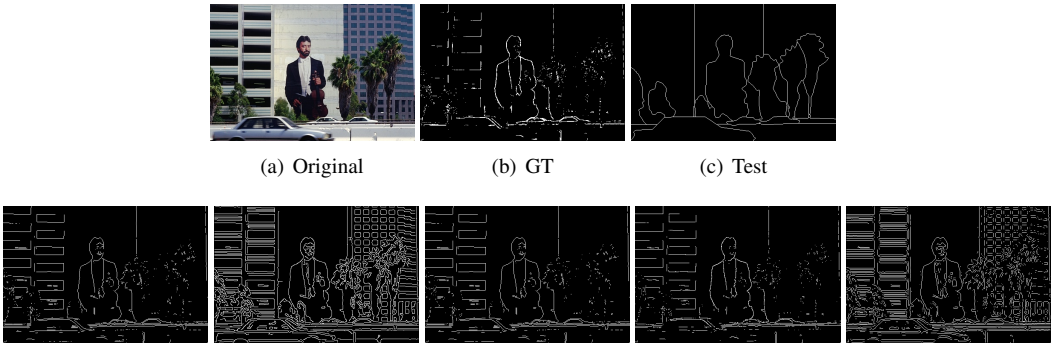
Figure B.18 Image 103070



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

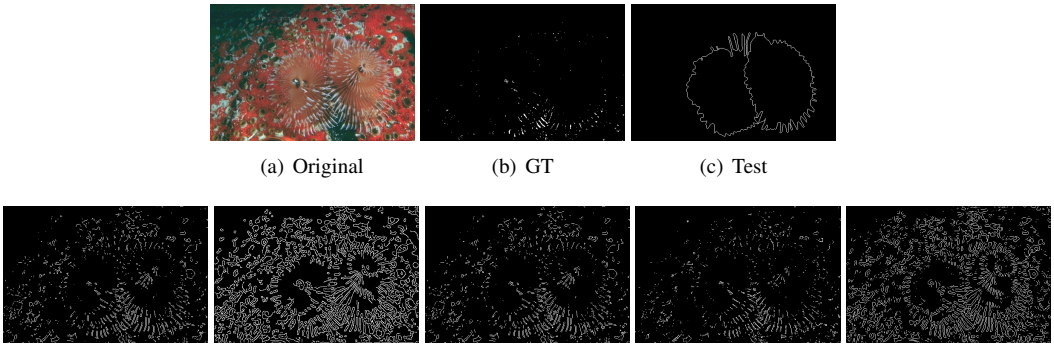
Figure B.19 Image 119082



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

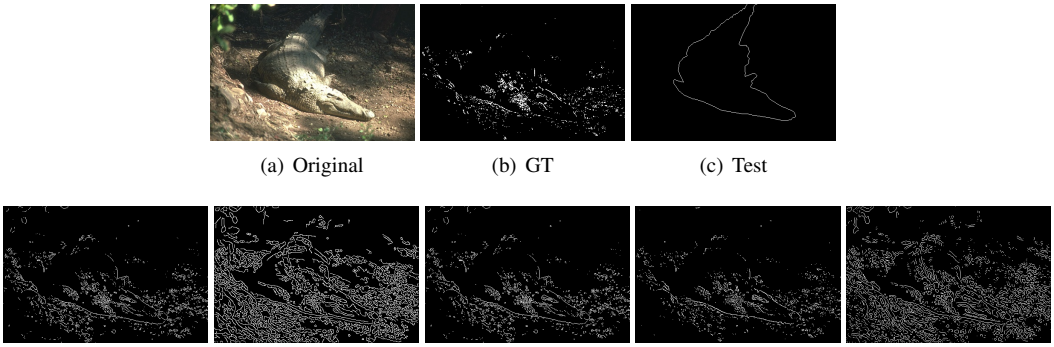
Figure B.20 Image 12084



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

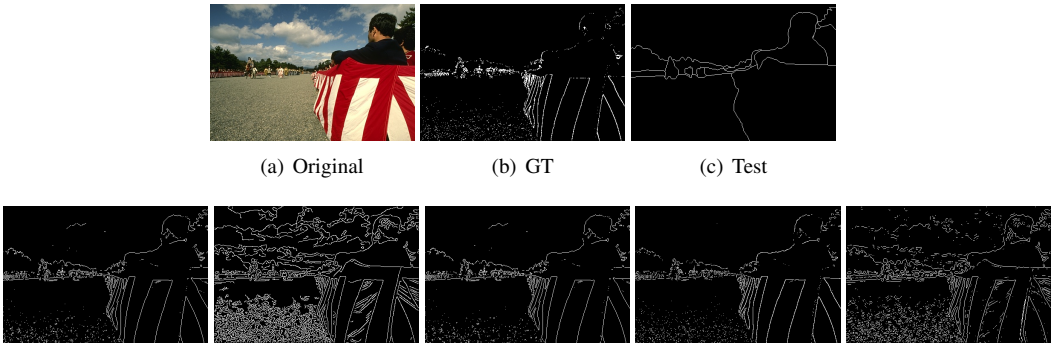
Figure B.21 Image 130026



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

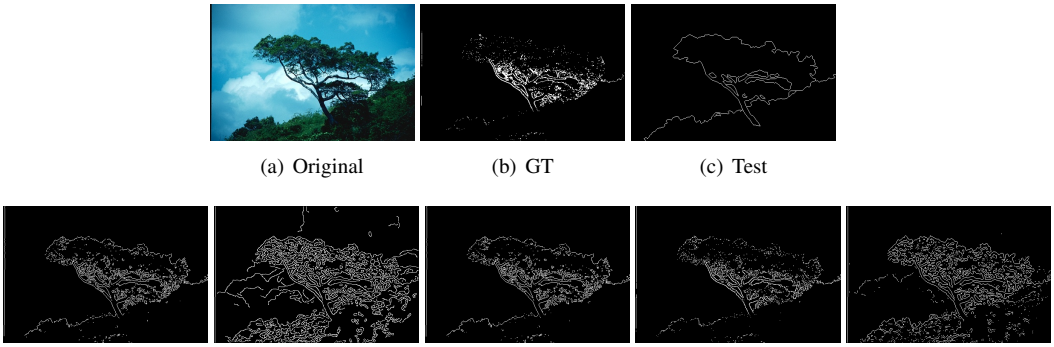
Figure B.22 Image 145086



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

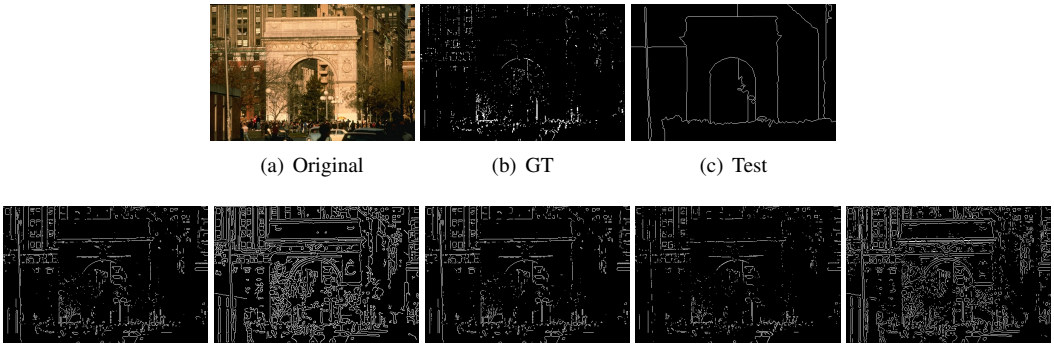
Figure B.23 Image 147091



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

Figure B.24 Image 148089



(a) Original (b) GT (c) Test

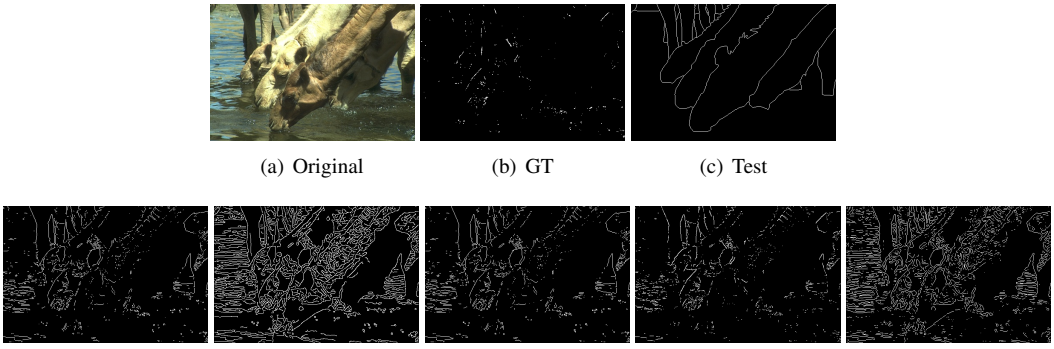
(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

Figure B.25 Image 157055

(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

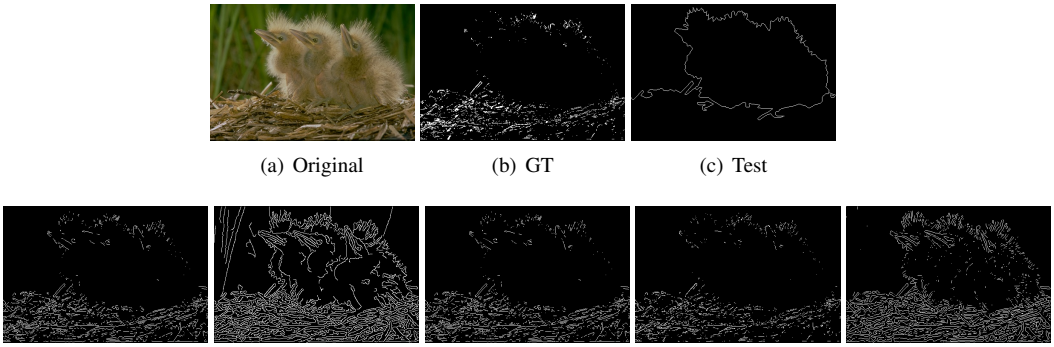
Figure B.26 Image 16077



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

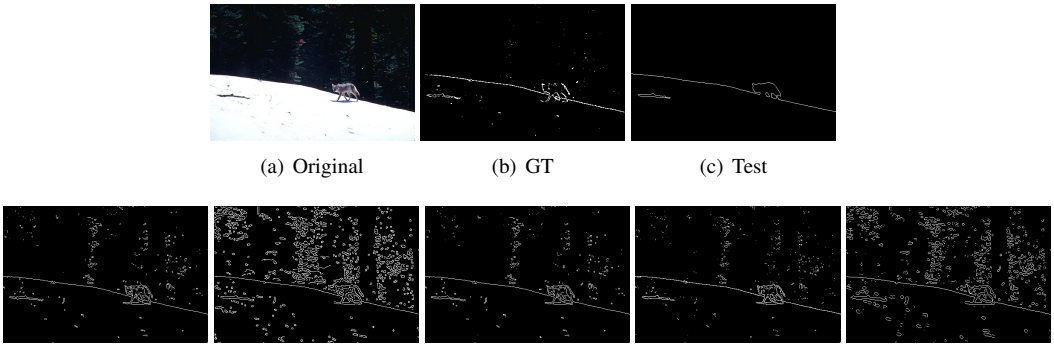
Figure B.27 Image 163085



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

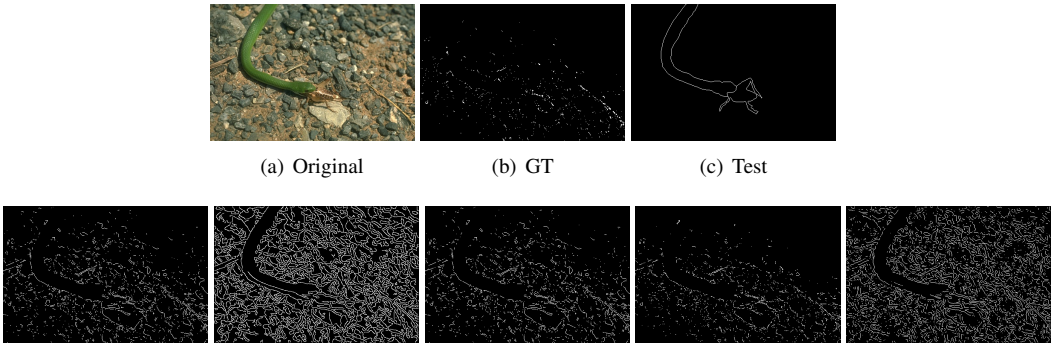
Figure B.28 Image 167062



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

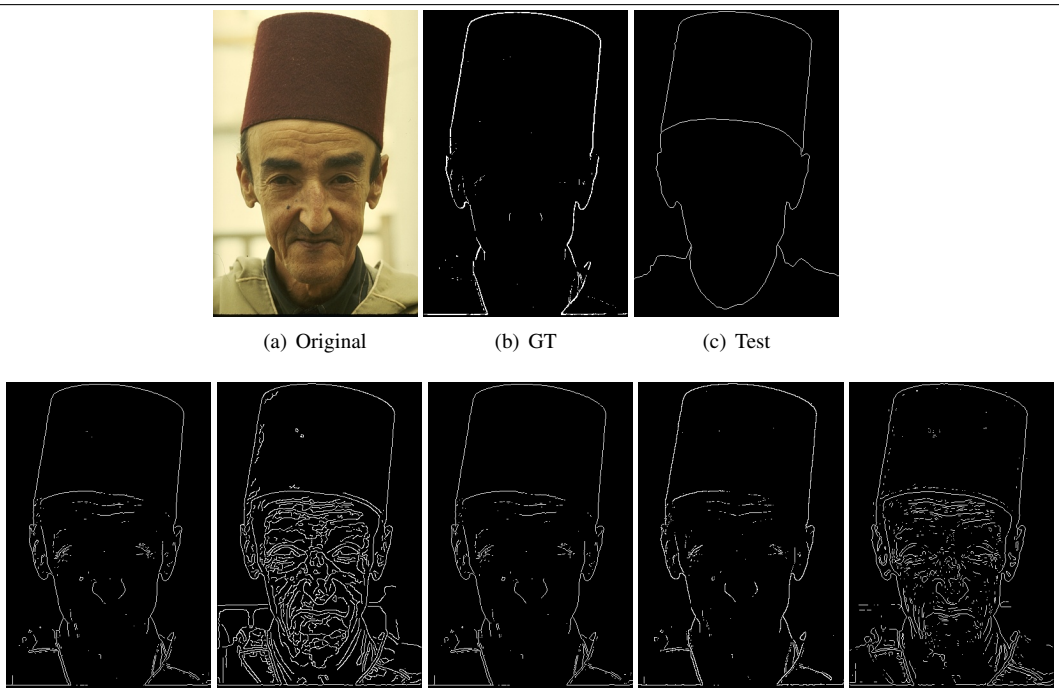
Figure B.29 Image 175043



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

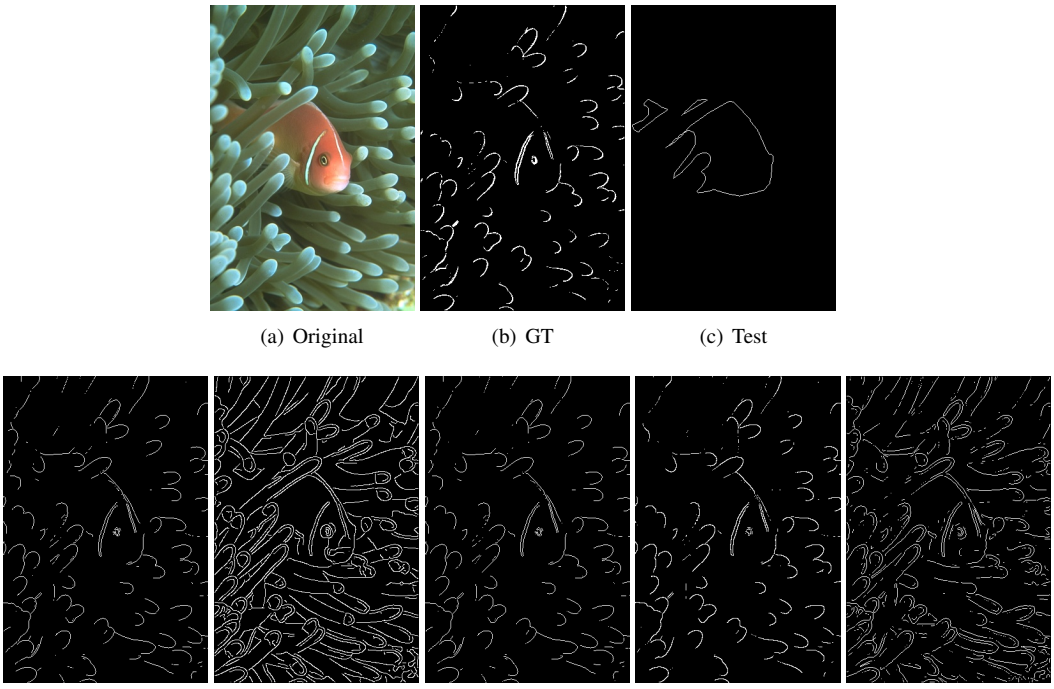
Figure B.30 Image 189080



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

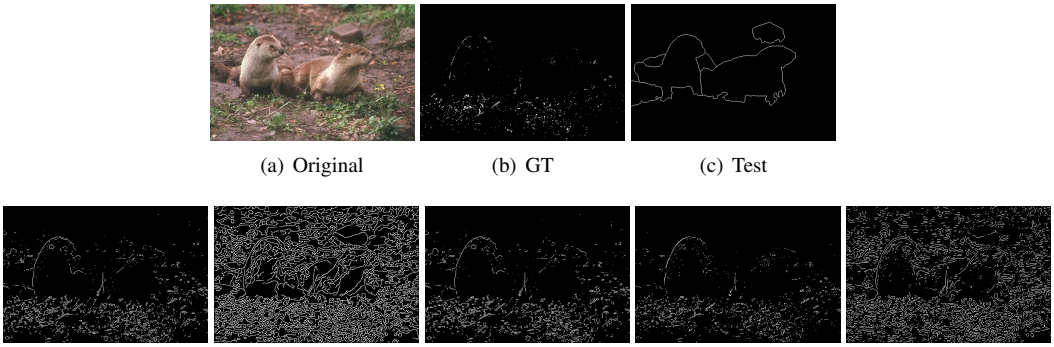
Figure B.31 Image 210088



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

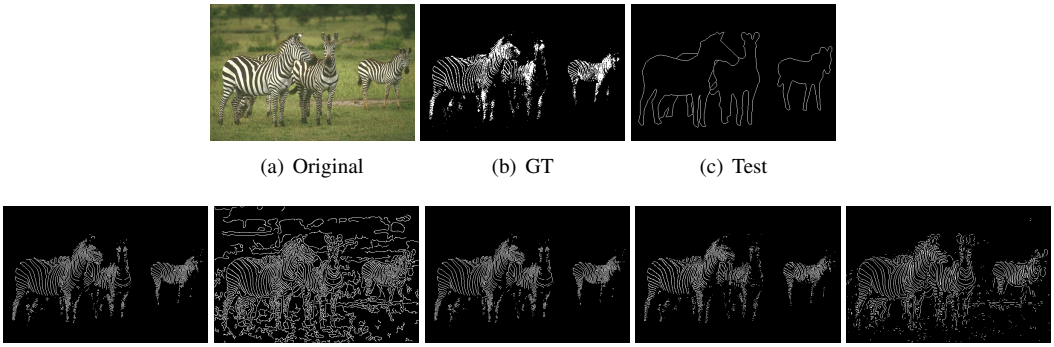
Figure B.32 Image 236037



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

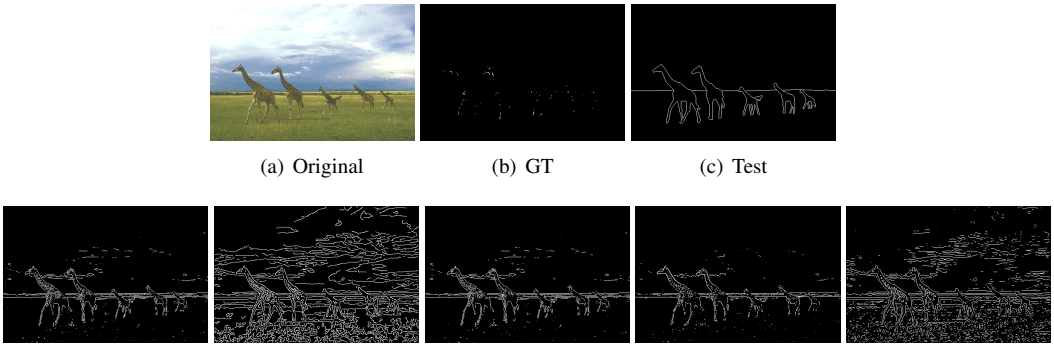
Figure B.33 Image 253027



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

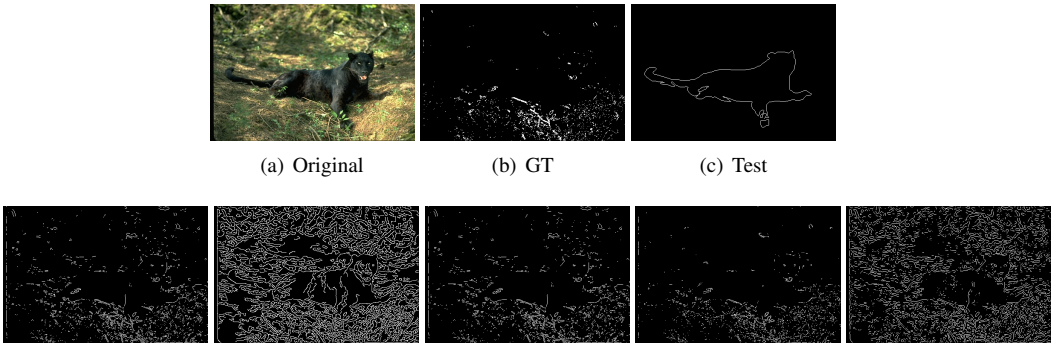
Figure B.34 Image 253055



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

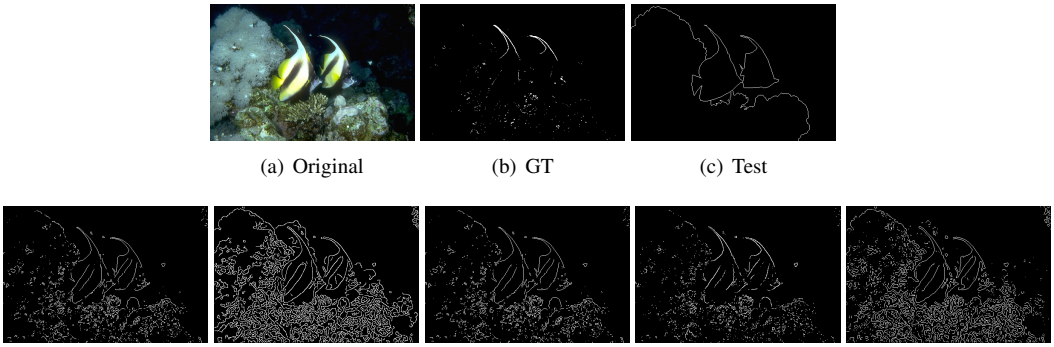
Figure B.35 Image 304034



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

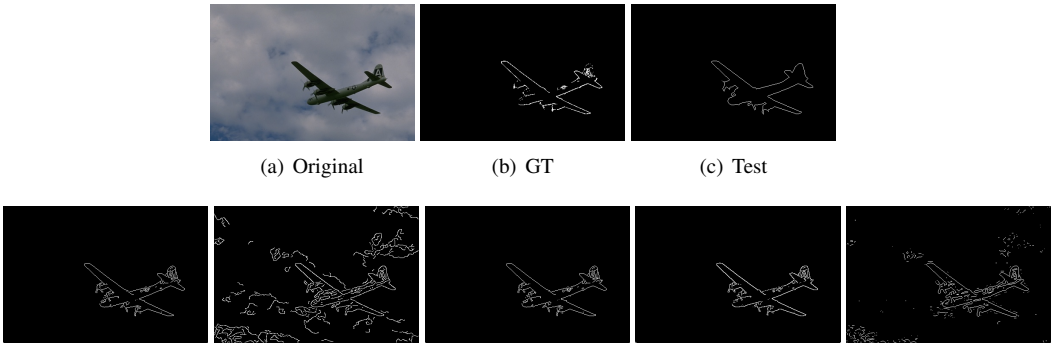
Figure B.36 Image 306005



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

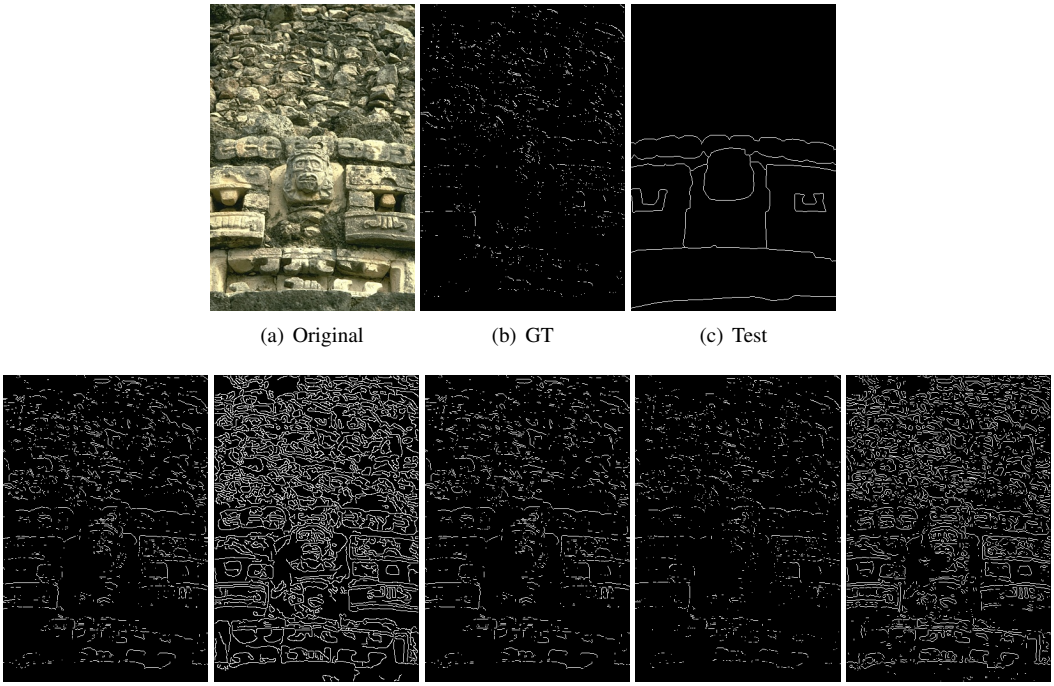
Figure B.37 Image 3096



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

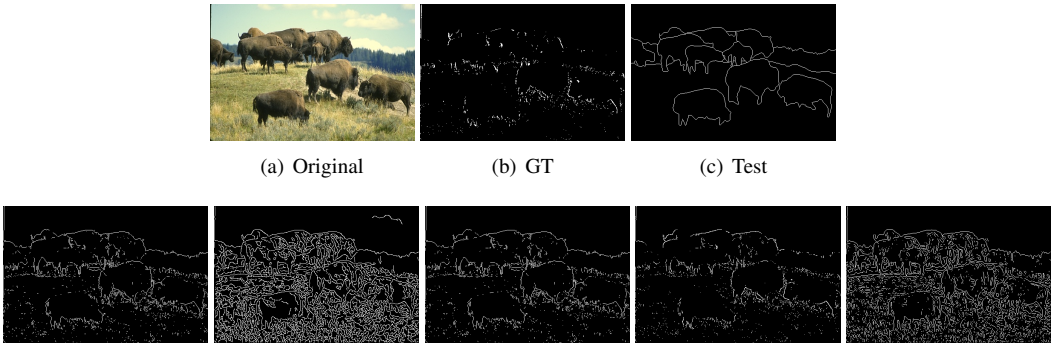
Figure B.38 Image 33039



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

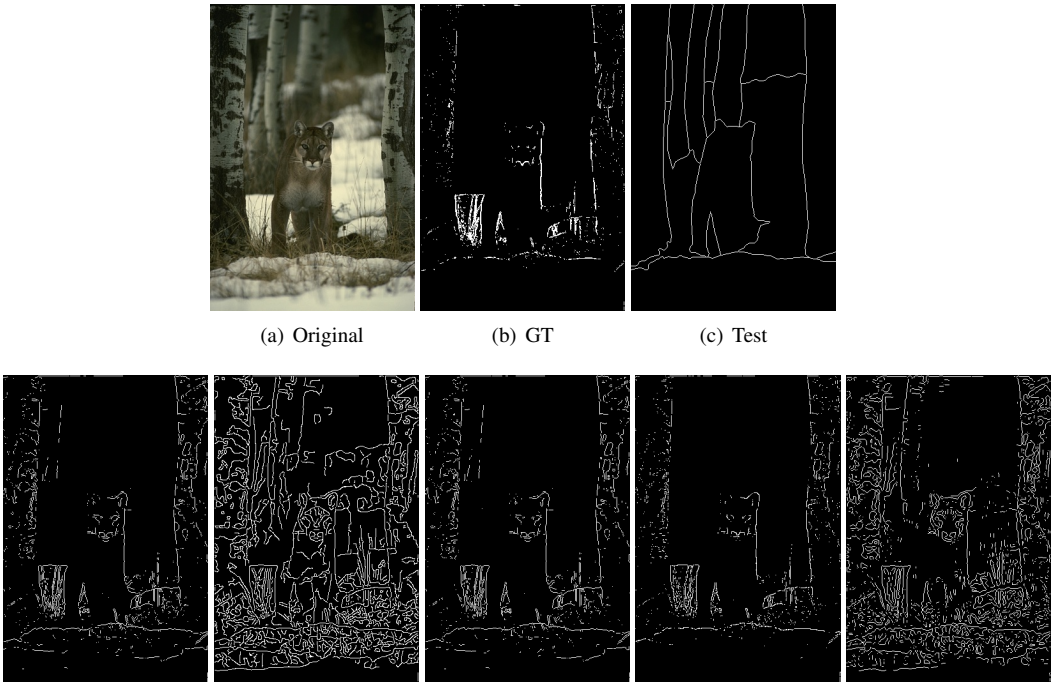
Figure B.39 Image 38092



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

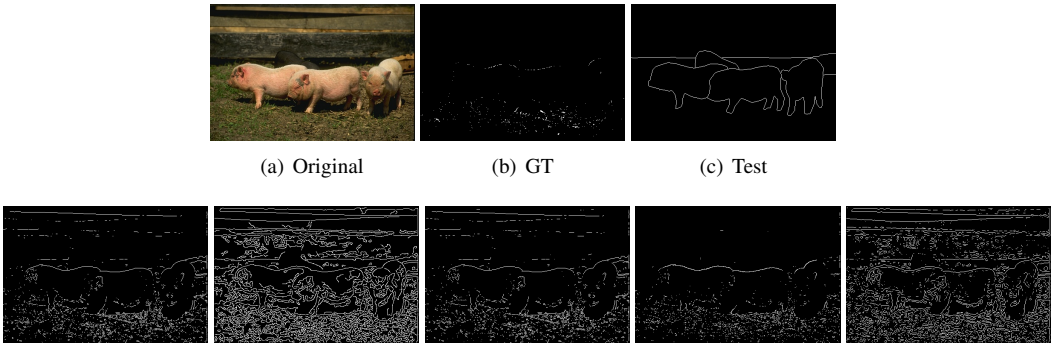
Figure B.40 Image 42012



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

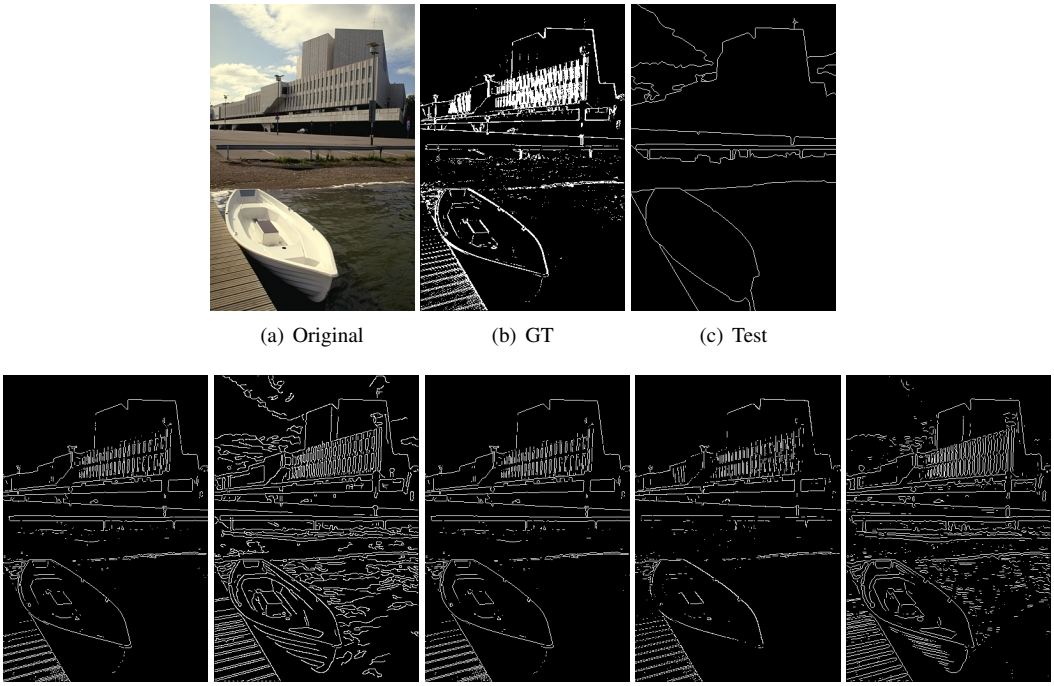
Figure B.41 Image 66053



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

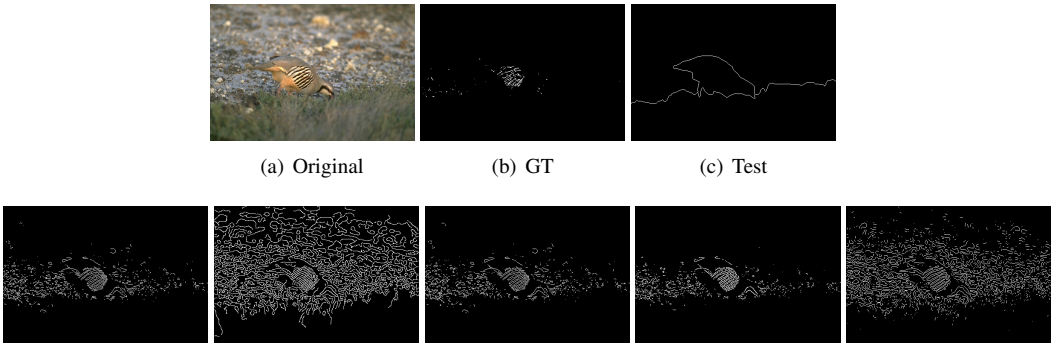
Figure B.42 Image 78004



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

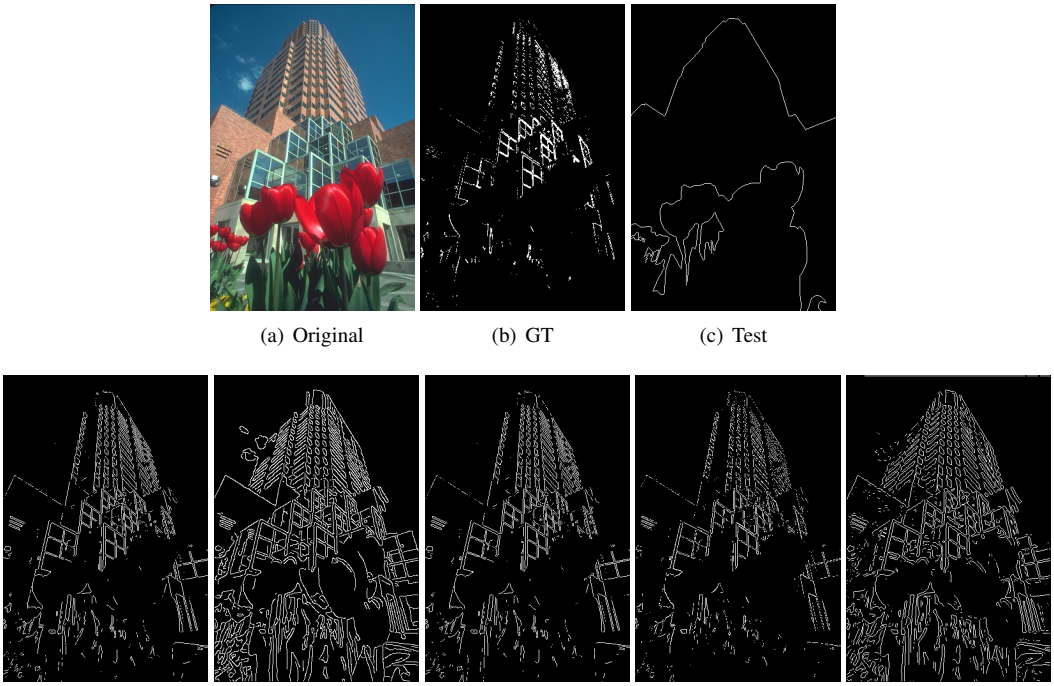
Figure B.43 Image 8023



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

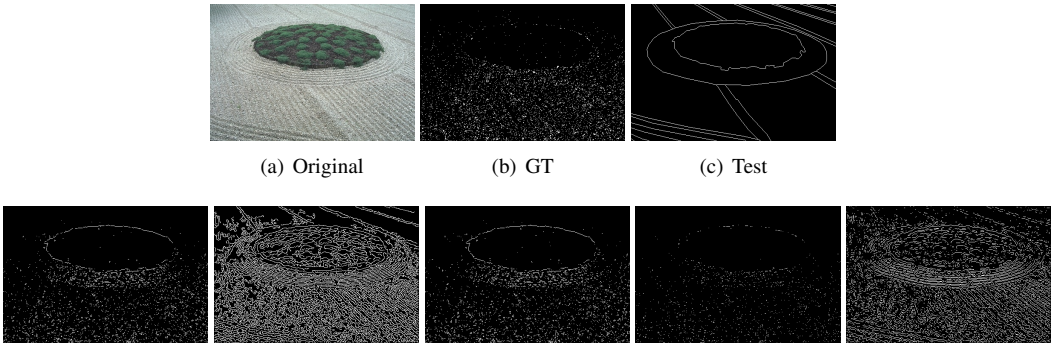
Figure B.44 Image 86000



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

Figure B.45 Image 86016

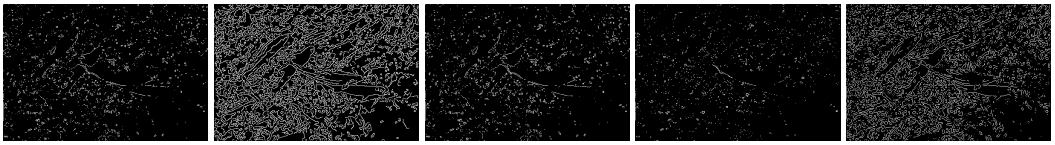


(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

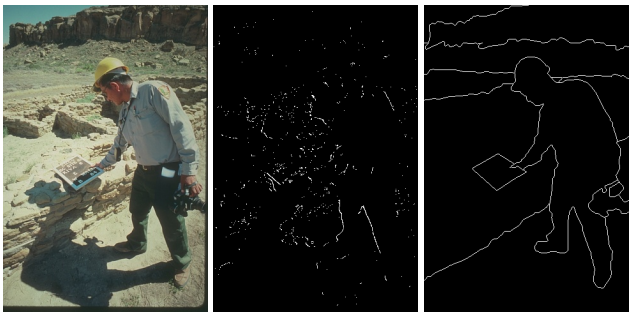
Figure B.46 Image 86068

(a) Original (b) GT (c) Test

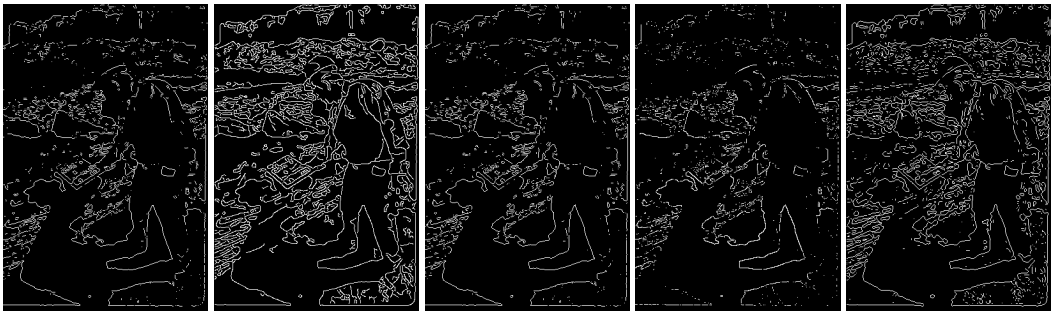


(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

Figure B.47 Image 89072

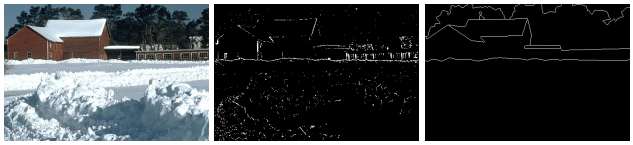


(a) Original (b) GT (c) Test



(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

Figure B.48 Image 97033



(a) Original (b) GT (c) Test

(d) Sobel (e) Canny (f) Prewitt (g) Roberts (h) LoG

Appendix C

Successful images from Third Run

C.1 Non-Fuzzy Sparse Busy image comparison

Table C.1 Busy Non-Fuzzy Run Successful Images with Filter and BDM for Comparison

Image	CA	Test	Sobel	Canny	Prewitt	Roberts	LoG	CVF
101087	15	12.952	24.815	14.439	24.813	25.01	25.65	24.73
103070	390	25.134	37.171	26.547	37.313	36.032	30.11	37.37
123074	259	72.989	74.487	75.029	74.317	74.682	73.99	73.61
14037	137	31.702	36.377	40.459	36.53	38.804	40.99	27.34
163085	236	38.532	40.131	47.367	40.146	39.851	43	40.27
167062	200	87.465	111.42	122.08	110.95	113.71	121.02	112.77
220075	39	27.249	29.767	30.203	29.695	29.181	34.89	33.53
236037	7	23.928	25.302	24.196	24.748	36.924	23.37	23.1
253055	324	25.049	67.904	97.697	67.588	55.118	95.64	51.9
260058	422	86.414	101.4	103.27	101.39	101.06	103.19	96.5
3096	364	21.939	64.911	45.104	64.922	64.309	40.13	64.53
41033	75	40.418	42.876	50.185	42.691	43.816	49.66	42.51
42012	11	19.137	28.338	21.554	28.22	32.885	23.32	27
43074	456	30.572	52.108	62.436	52.29	52.685	55.5	52.45
62096	365	42.365	46.063	49.273	46.331	46.85	48.6	47.74
69040	230	44.117	44.434	45.099	44.564	43.564	44.74	44.35
86068	387	72.325	74.582	79.424	73.9	70.819	79.27	74.5
87046	420	17.783	54.445	24.189	54.436	55.086	23.66	26.65

Table C.2 Sparse Non-Fuzzy Run Successful Images with Filter and BDM for Comparison

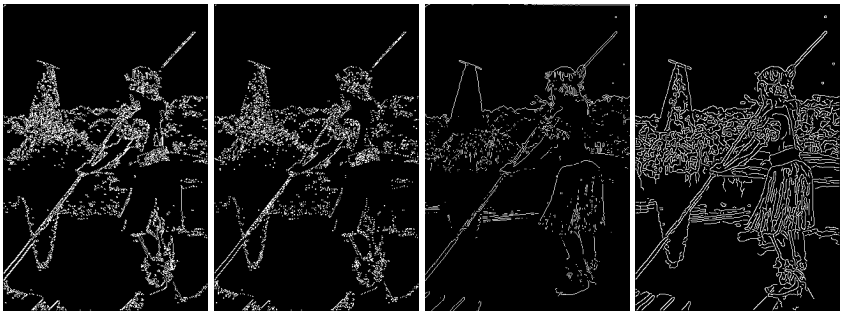
Image	CA	Test	Sobel	Canny	Prewitt	Roberts	LoG	CVF
101087	47	18.158	26.296	19.559	26.294	26.083	28.881	27.161
105025	416	62.171	67.877	73.262	67.729	65.161	72.109	70.905
106024	73	45.537	46.697	103.46	46.647	61.373	70.676	46.062
123074	3	72.272	73.966	74.207	73.793	74.131	73.284	73.012
143090	11	55.437	68.257	83.887	68.311	67.143	80.475	68.043
157055	39	32.391	34.959	42.029	35.142	34.185	41.213	39.13
167062	267	89.919	114.79	126.43	114.39	117.73	125.23	116.2
227092	15	37.818	38.227	43.719	38.066	38.471	42.527	34.093
253055	100	25.363	68.083	98.054	67.73	55.266	96.01	18.176
260058	228	98.783	118.39	123.14	118.32	116.62	122.62	111.29
296007	484	37.754	38.287	42.206	38.471	38.929	66.562	41.283
296059	110	27.162	28.895	35.789	28.837	27.686	35.178	27.185
41033	15	42.199	45.059	53.237	44.916	45.246	52.674	44.749
42012	388	26.772	33.773	32.947	33.422	33.35	33.603	34.408
43074	207	56.988	69.727	80.899	69.833	69.58	75.229	69.31
62096	267	64.687	68.495	71.03	68.646	68.991	70.532	70.097
85048	207	31.287	33.115	35.878	33.044	32.431	35.567	35.82
86068	3	113.78	115.48	120.23	114.98	112.42	120.04	115.6

Table C.3 Non-Fuzzy Run Successful Image for Both Systems with Filter and BDM for Comparison

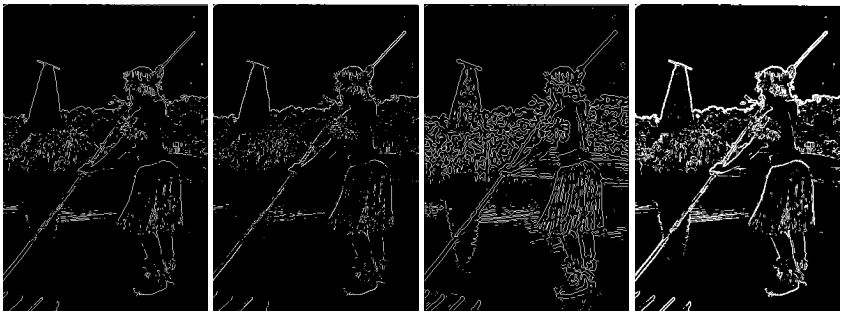
Image	Filter Sparse	BDM Sparse	Filter Busy	BDM Busy
101087	47	18.158	15	12.952
123074	3	72.272	259	72.989
167062	267	89.919	200	87.465
253055	100	25.363	324	25.049
260058	228	98.783	422	86.414
41033	15	42.199	75	40.418
42012	388	26.772	11	19.137
43074	207	56.988	456	30.572
62096	267	64.687	365	42.365
86068	3	113.78	387	72.325

Figure C.1 Images 101087

(a) Original Image (b) Sparse GT (c) Busy GT



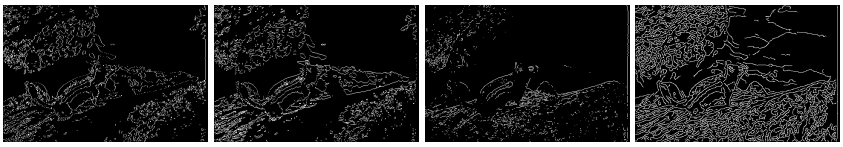
(d) Sparse (e) Busy (f) Sobel (g) Canny



(h) Prewitt (i) Roberts (j) LoG (k) CVF

Figure C.2 Images 123074

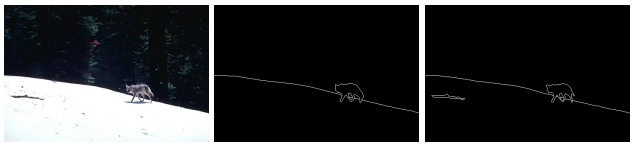
(a) Original Image (b) Sparse GT (c) Busy GT



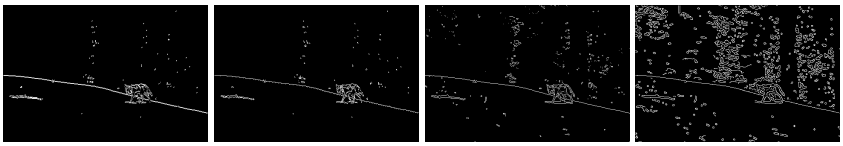
(d) Sparse (e) Busy (f) Sobel (g) Canny

(h) Prewitt (i) Roberts (j) LoG (k) CVF

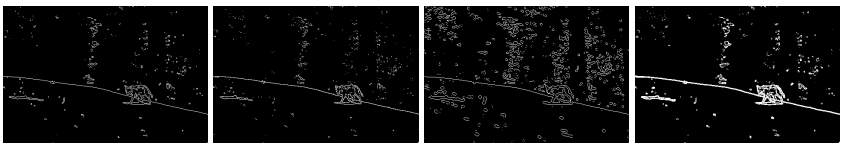
Figure C.3 Images 167062



(a) Original Image (b) Sparse GT (c) Busy GT

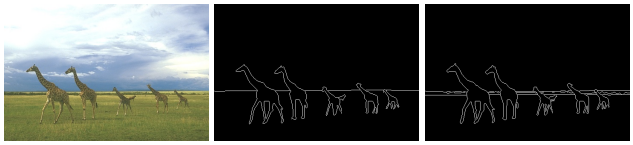


(d) Sparse (e) Busy (f) Sobel (g) Canny



(h) Prewitt (i) Roberts (j) LoG (k) CVF

Figure C.4 Images 253055

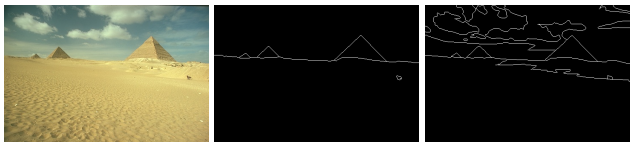


(a) Original Image (b) Sparse GT (c) Busy GT

(d) Sparse (e) Busy (f) Sobel (g) Canny

(h) Prewitt (i) Roberts (j) LoG (k) CVF

Figure C.5 Images 260058



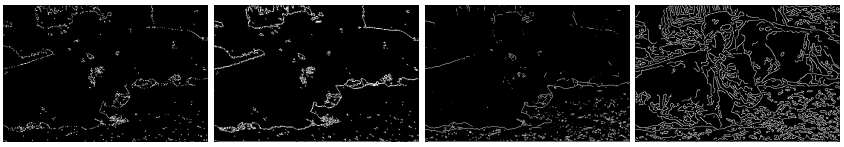
(a) Original Image (b) Sparse GT (c) Busy GT

(d) Sparse (e) Busy (f) Sobel (g) Canny

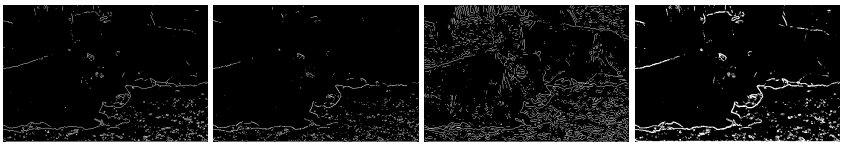
(h) Prewitt (i) Roberts (j) LoG (k) CVF

Figure C.6 Images 41033

(a) Original Image (b) Sparse GT (c) Busy GT

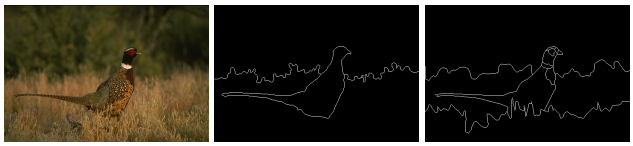


(d) Sparse (e) Busy (f) Sobel (g) Canny

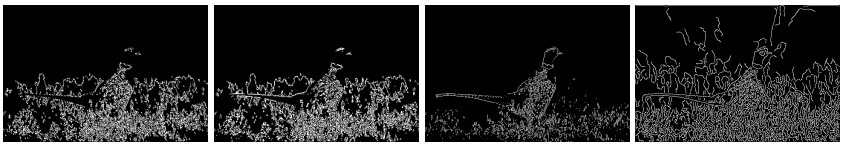


(h) Prewitt (i) Roberts (j) LoG (k) CVF

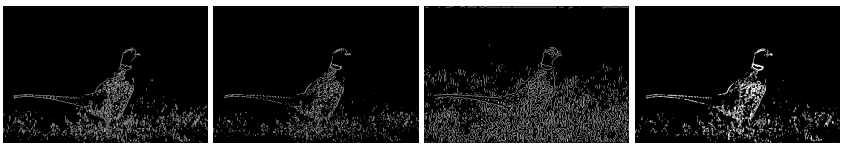
Figure C.7 Images 43074



(a) Original Image (b) Sparse GT (c) Busy GT

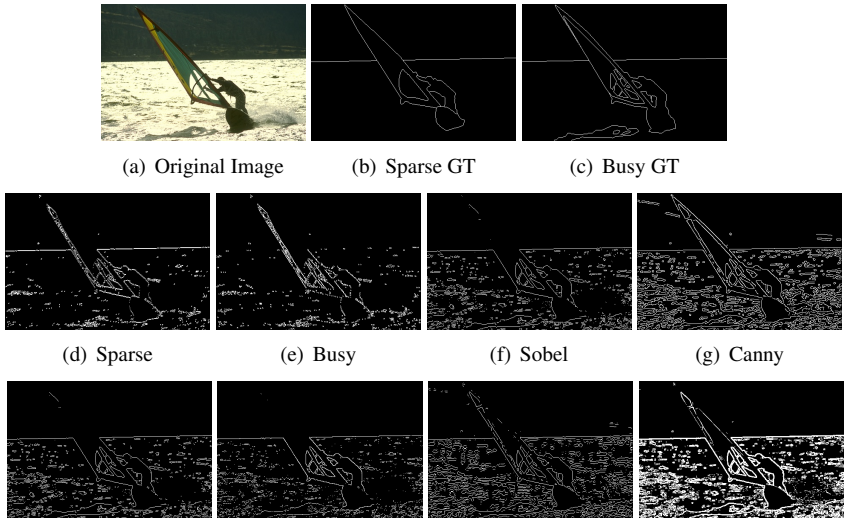


(d) Sparse (e) Busy (f) Sobel (g) Canny



(h) Prewitt (i) Roberts (j) LoG (k) CVF

Figure C.8 Images 62096

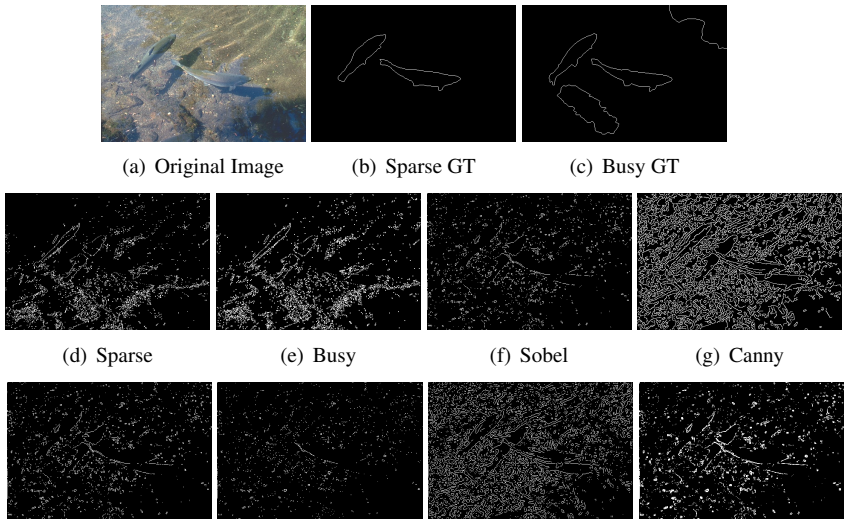


(a) Original Image (b) Sparse GT (c) Busy GT

(d) Sparse (e) Busy (f) Sobel (g) Canny

(h) Prewitt (i) Roberts (j) LoG (k) CVF

Figure C.9 Images 86068



(a) Original Image (b) Sparse GT (c) Busy GT

(d) Sparse (e) Busy (f) Sobel (g) Canny

(h) Prewitt (i) Roberts (j) LoG (k) CVF

Figure C.10 Images 42012

(a) Original Image (b) Sparse GT (c) Busy GT

(d) Sparse (e) Busy (f) Sobel (g) Canny

(h) Prewitt (i) Roberts (j) LoG (k) CVF

C.2 Fuzzy Sparse Busy image comparison

Table C.4 Busy Fuzzy System: Filters Selected by Training Processes

OffSet	Threshold	CA
124	0.67	203
82	0.712	178
97	0.73	56
66	0.53599	98
51	0.23	123
92	0.6	326
73	0.43	168
43	0.35611	280
65	0.49631	13
101	0.34	121
68	0.47	386
39	0.4	45
71	0.19	355
60	0.3	23
134	0.3	18
35	0.71	45
112	0.54	410
86	0.87	245
3	0.84878	67
38	0.47601	262
31	0.21954	41

Table C.5 Busy Fuzzy System: Successful Images with Filter and BDM for Comparison

Image	OffSet	Threshold	CA	Test	Sobel	Canny	Prewitt	Robert	LoG	CFV
108070	82	0.712	178	46.321	51.969	70.034	51.516	47.116	69.5	65.6
108082	92	0.6	326	40.028	44.39	56.527	44.268	40.164	56.1	53.4
12084	82	0.712	178	45.881	75.692	84.451	75.544	72.047	84	81.9
123074	92	0.6	326	73.076	74.487	75.029	74.317	74.682	71	73.6
14037	66	0.53599	98	34.592	36.377	40.459	36.53	38.804	41	27.3
167062	82	0.712	178	93.249	111.42	122.08	110.95	113.71	121	112.8
196073	82	0.712	178	92.159	116	133.16	114.78	109.57	132.4	106.7
210088	112	0.54	410	45.276	49.342	54.505	48.612	48.499	53.4	51.2
253058	134	0.3	18	25.408	67.904	97.697	67.588	55.118	95.6	51.9
304034	97	0.73	56	52.252	78.965	87.108	79.34	68.253	86.7	86.5
69020	82	0.712	178	31.982	40.964	45.443	40.807	38.932	45.1	42.4
8023	92	0.6	326	32.68	51.456	91.748	51.592	34.752	88.8	68.9
86068	97	0.73	56	48.117	74.582	79.424	73.9	70.819	79.2	74.5

Table C.6 Sparse Fuzzy System: Filters Selected by Training Processes

OffSet	Threshold	CA
82	0.712	178
92	0.6	326
112	0.54	410
97	0.73	56
73	0.43	168
86	0.87	245
39	0.4	45
60	0.3	23
35	0.71	45
134	0.3	18
101	0.34	121
36	0.46386	368
71	0.19	355
68	0.47	386
124	0.67	203
9	0.34753	3

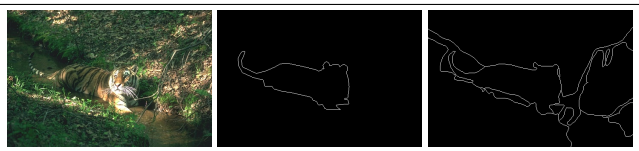
Table C.7 Sparse Fuzzy System: Successful Images With Filter and BDM for Comparison

Image	OffSet	Threshold	CA	Test	Sobel	Canny	Prewitt	Robert	LoG	CVF
105025	82	0.712	178	47.957	67.877	73.262	67.729	65.161	72.1	70.9
108070	97	0.73	56	95.497	102.66	119.04	102.27	99.548	118.8	115.1
108082	82	0.712	178	34.174	45.798	58.844	45.818	38.318	58.3	55.4
109053	82	0.712	178	47.909	57.805	65.194	57.91	53.759	63.5	59.2
119082	82	0.712	178	21.846	26.432	31.496	26.519	26.376	30.8	31.5
12084	82	0.712	178	45.052	75.557	84.516	75.378	71.856	84.1	81.9
126007	82	0.712	178	36.973	50.285	58.484	49.698	45.205	59.5	51.6
130026	82	0.712	178	75.696	87.128	93.264	85.671	80.547	93.3	91.6
14037	35	0.71	45	59.698	67.171	76.702	67.253	63.315	76.4	66.9
143090	82	0.712	178	30.115	68.257	83.887	68.311	67.143	80.5	68
145086	82	0.712	178	69.121	74.127	84.59	73.736	72.295	83	79.4
148089	82	0.712	178	44.4	45.994	49.866	45.911	45.302	49.7	50.2
156065	82	0.712	178	134.87	141.81	146.96	141.78	136.98	146.6	145.3
157055	82	0.712	178	32.918	34.959	42.029	35.142	34.185	41.2	39.1
159008	82	0.712	178	206.34	221.12	225.77	221.09	218.49	226	225.3
16077	82	0.712	178	35.613	42.267	45.941	42.233	41.831	46	44.3
163085	92	0.6	326	40.413	44.76	58.329	44.754	43.894	54.8	45.3
167062	82	0.712	178	97.042	114.79	126.43	114.39	117.73	125.2	116.2
175043	97	0.73	56	78.422	120.11	128.73	120.66	100.98	128.2	128.3
196073	97	0.73	56	77.144	140.63	156.03	139.54	135.03	155.2	132.2
197017	82	0.712	178	35.387	40.2	59.349	40.102	40.143	53.5	41
208001	92	0.6	326	60.087	71.978	82.198	71.66	63.161	81.9	74.9
236037	82	0.712	178	54.863	62.418	71.959	62.232	59.339	71.2	69.9
253027	82	0.712	178	14.323	15.914	46.772	16.036	14.813	25.4	18.2
253055	97	0.73	56	43.033	68.083	98.054	67.73	55.266	96	52.1
260058	82	0.712	178	89.913	118.39	123.14	118.32	116.62	122.6	111.3
271035	82	0.712	178	36.419	48.68	52.957	48.615	46.104	52.9	51.9
300091	82	0.712	178	136.91	142.16	150.46	141.85	141.83	151.3	143.4
304034	97	0.73	56	55.963	83.486	91.625	83.826	73.122	91.1	91
304074	82	0.712	178	56.655	63.221	72.387	62.374	62.453	71.8	71
306005	82	0.712	178	61.829	72.744	76.468	72.571	72.217	79.3	74.5
3096	92	0.6	326	2.8561	45.142	115.97	45.142	45.445	107.9	44.4
361010	82	0.712	178	46.887	48.834	57.844	49.121	49.128	55.2	49.2
37073	82	0.712	178	60.267	93.076	98.537	93.067	92.859	98.5	93.2
376043	82	0.712	178	36.924	51.455	62.333	51.351	47.037	62.2	60.7
38092	82	0.712	178	30.358	33.593	41.557	33.381	32.832	36.7	35.9
41069	82	0.712	178	137.87	144.54	149.18	144.25	138.04	148.8	148.6
54082	73	0.43	168	27.717	28.148	32.027	28.405	28.002	31.2	29.9
65033	82	0.712	178	58.073	61.273	68.804	61.281	58.341	68.2	66.9
66053	82	0.712	178	37.687	64.588	69.003	64.794	44.601	68.9	67
69020	82	0.712	178	52.575	62.839	67.007	62.62	60.997	66.7	64.1
69040	82	0.712	178	125.82	134.58	142.86	134.6	130.87	142.2	141
76053	92	0.6	326	60.337	64.692	68.35	64.373	62.5	68.2	65
8023	35	0.71	45	89.232	102.3	139.02	102.02	91.267	137.7	117.5
86000	82	0.712	178	32.44	34.788	38.428	34.834	34.155	43.7	42.6
86068	82	0.712	178	108.69	115.48	120.23	114.98	112.42	120	115.6
89072	82	0.712	178	28.877	35.644	37.535	35.581	35.295	37.7	37.3

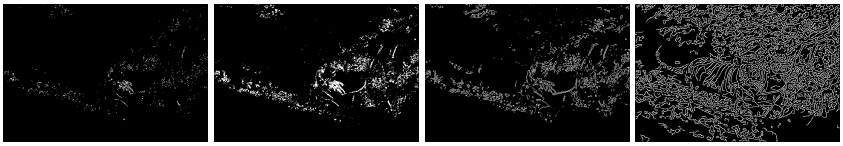
Table C.8 Images Which Both Sparse and Busy Systems Solved Better Than Standard Methods

Image	Sparse Filter	Sparse BDM	Busy Filter	Busy BDM
108070	97 , 0.73 , 56	95.497	82 , 0.712 , 178	46.321
108082	82 , 0.712 , 178	34.174	92 , 0.6 , 326	40.028
12084	82 , 0.712 , 178	45.052	82 , 0.712 , 178	45.881
14037	35 , 0.71 , 45	59.698	66 , 0.53599 , 98	34.592
167062	82 , 0.712 , 178	97.042	82 , 0.712 , 178	93.249
196073	97 , 0.73 , 56	77.144	82 , 0.712 , 178	92.159
304034	97 , 0.73 , 56	55.963	97 , 0.73 , 56	52.252
69040	82 , 0.712 , 178	125.82	82 , 0.712 , 178	31.982
8023	35 , 0.71 , 45	89.232	92 , 0.6 , 326	32.68
86068	82 , 0.712 , 178	108.69	97 , 0.73 , 56	48.117

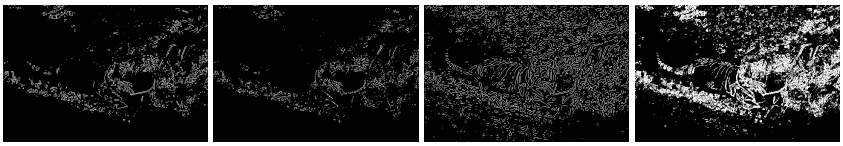
Figure C.11 Images 108070



(a) Original (b) Sparse GT (c) Busy GT

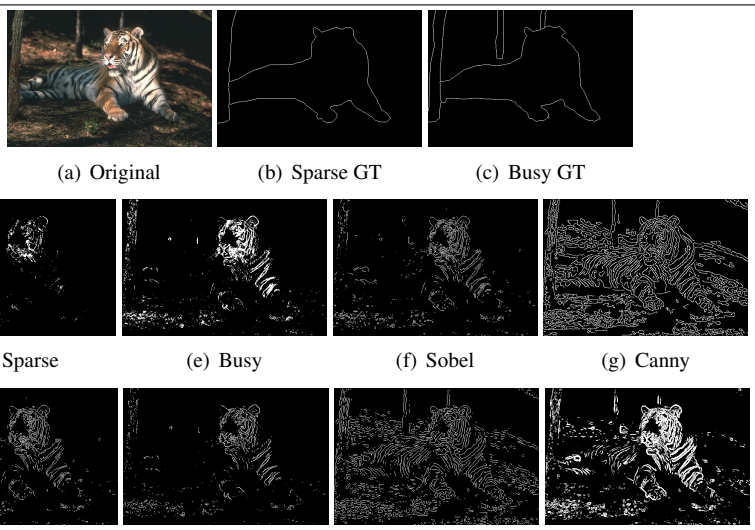


(d) Sparse (e) Busy (f) Sobel (g) Canny



(h) Prewitt (i) Roberts (j) LoG (k) CVF

Figure C.12 Images 108082

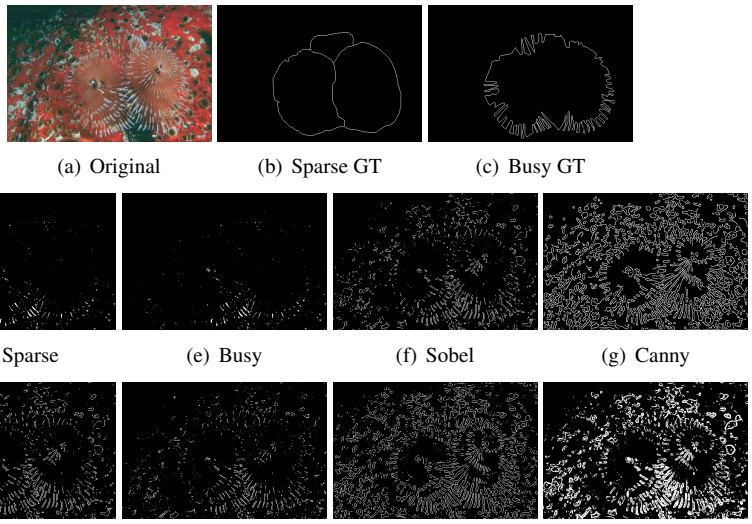


(a) Original (b) Sparse GT (c) Busy GT

(d) Sparse (e) Busy (f) Sobel (g) Canny

(h) Prewitt (i) Roberts (j) LoG (k) CVF

Figure C.13 Images 12084

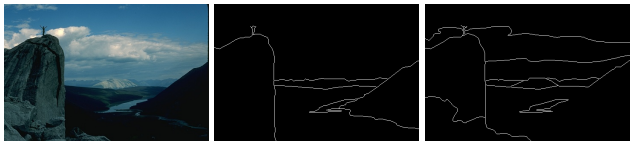


(a) Original (b) Sparse GT (c) Busy GT

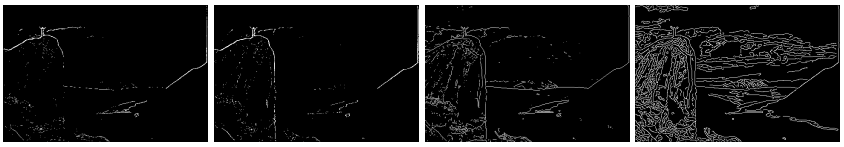
(d) Sparse (e) Busy (f) Sobel (g) Canny

(h) Prewitt (i) Roberts (j) LoG (k) CVF

Figure C.14 Images 14037



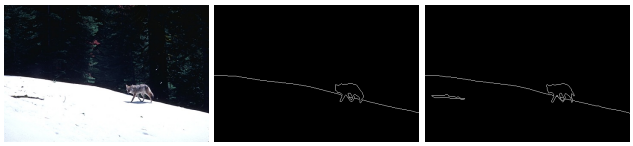
(a) Original (b) Sparse GT (c) Busy GT



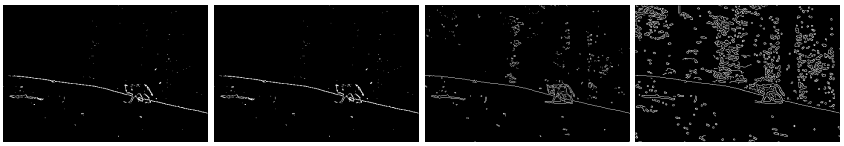
(d) Sparse (e) Busy (f) Sobel (g) Canny

(h) Prewitt (i) Roberts (j) LoG (k) CVF

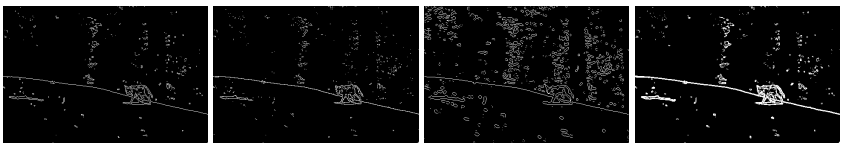
Figure C.15 Images 167062



(a) Original (b) Sparse GT (c) Busy GT

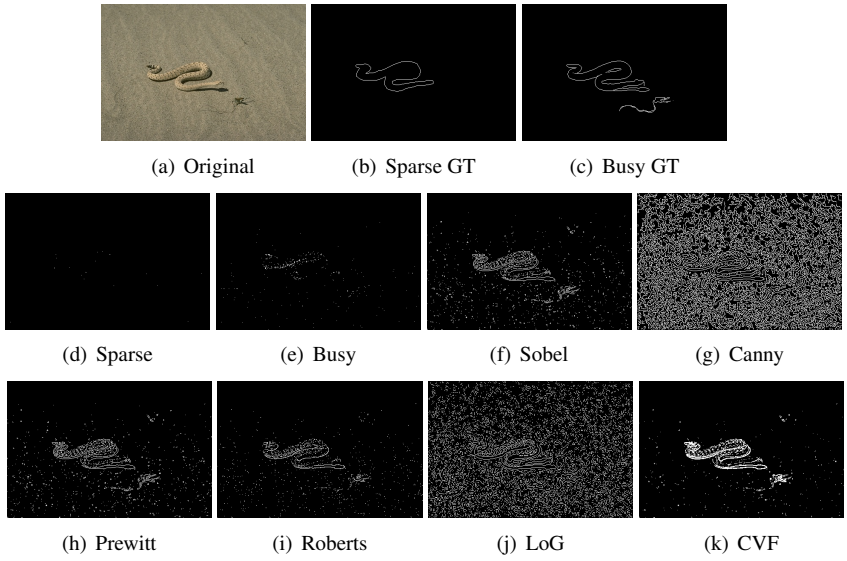


(d) Sparse (e) Busy (f) Sobel (g) Canny



(h) Prewitt (i) Roberts (j) LoG (k) CVF

Figure C.16 Images 196073

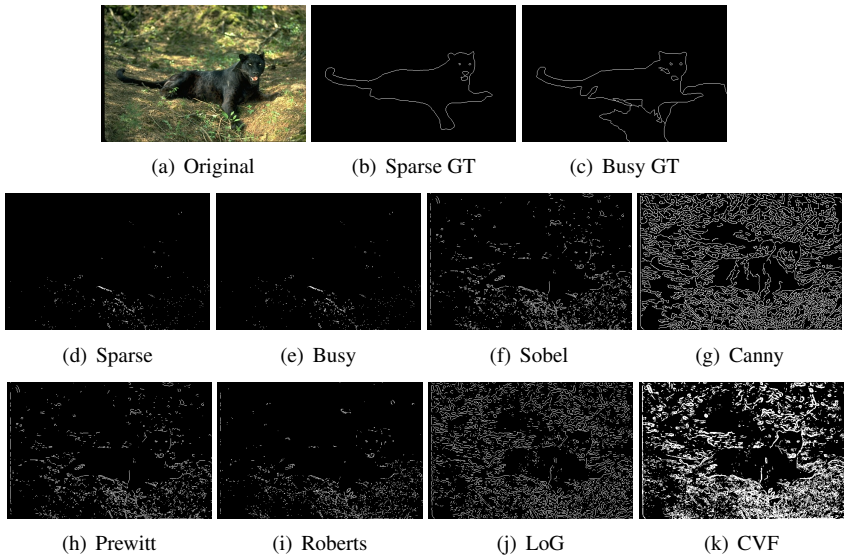


(a) Original (b) Sparse GT (c) Busy GT

(d) Sparse (e) Busy (f) Sobel (g) Canny

(h) Prewitt (i) Roberts (j) LoG (k) CVF

Figure C.17 Images 304034

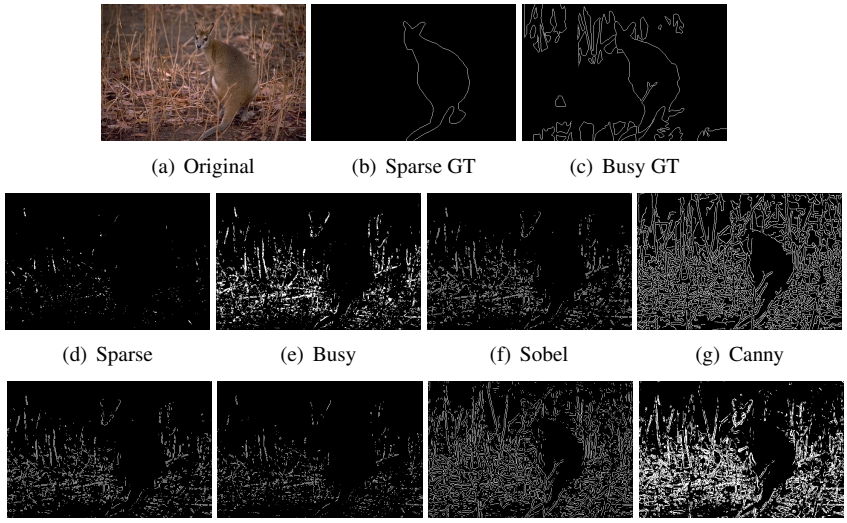


(a) Original (b) Sparse GT (c) Busy GT

(d) Sparse (e) Busy (f) Sobel (g) Canny

(h) Prewitt (i) Roberts (j) LoG (k) CVF

Figure C.18 Images 69040

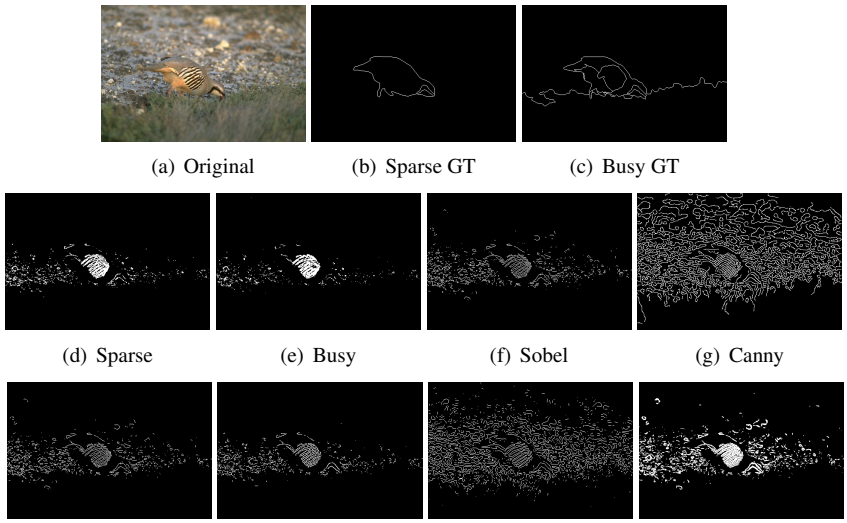


(a) Original (b) Sparse GT (c) Busy GT

(d) Sparse (e) Busy (f) Sobel (g) Canny

(h) Prewitt (i) Roberts (j) LoG (k) CVF

Figure C.19 Images 8023

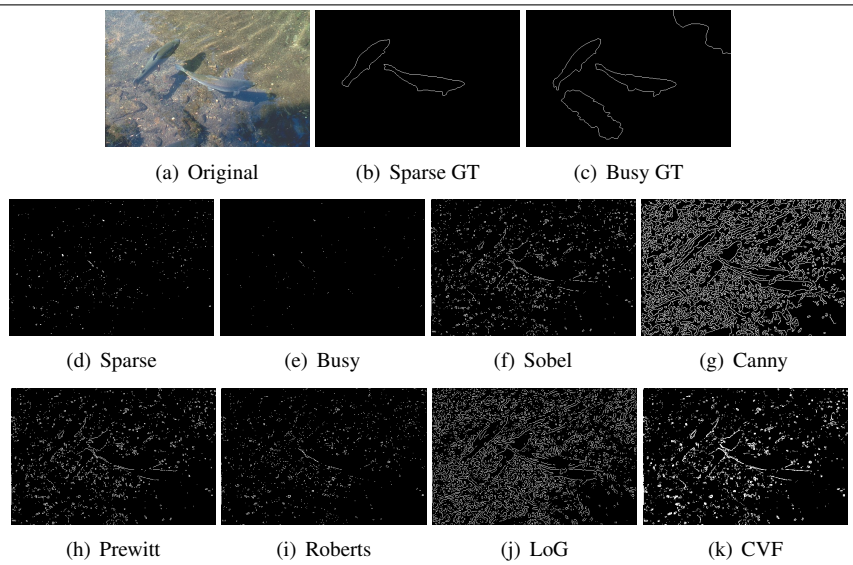


(a) Original (b) Sparse GT (c) Busy GT

(d) Sparse (e) Busy (f) Sobel (g) Canny

(h) Prewitt (i) Roberts (j) LoG (k) CVF

Figure C.20 Images 86068



(a) Original (b) Sparse GT (c) Busy GT

(d) Sparse (e) Busy (f) Sobel (g) Canny

(h) Prewitt (i) Roberts (j) LoG (k) CVF

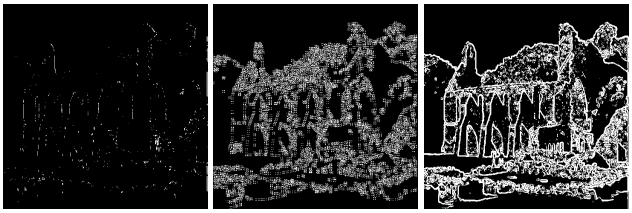
Appendix D

Images without Ground Truths

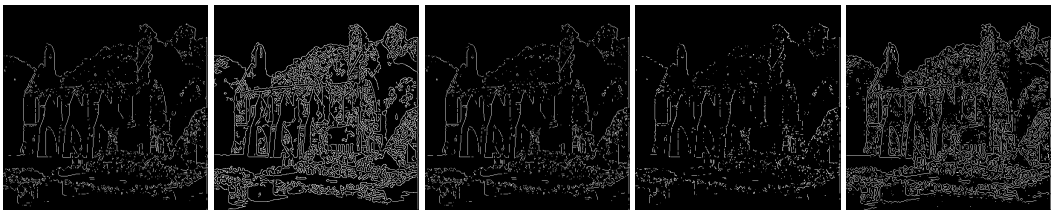
D.1 abbey

Figure D.1 Image aaimforvxlklilzm

(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



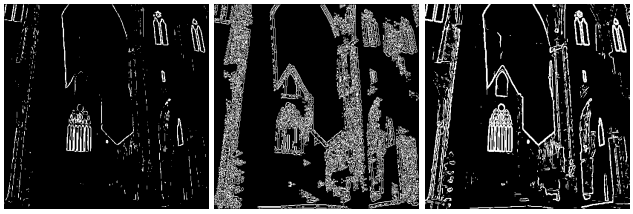
(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF



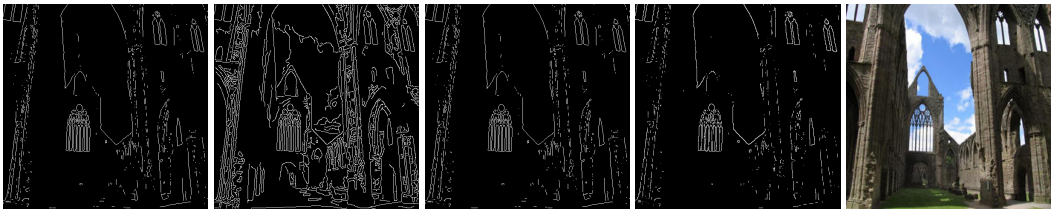
(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

Figure D.2 Image adsyiurcswacjxze

(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System

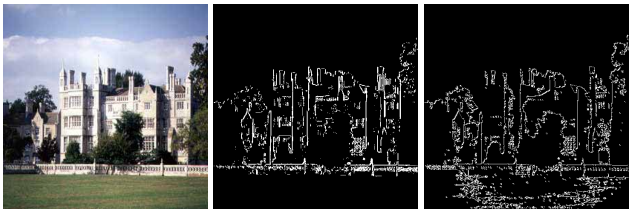


(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

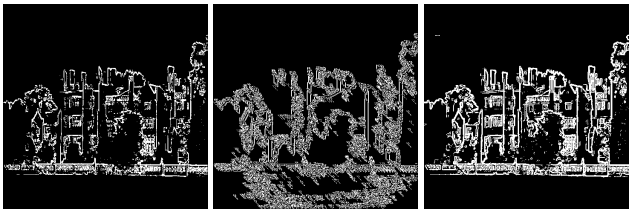


(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

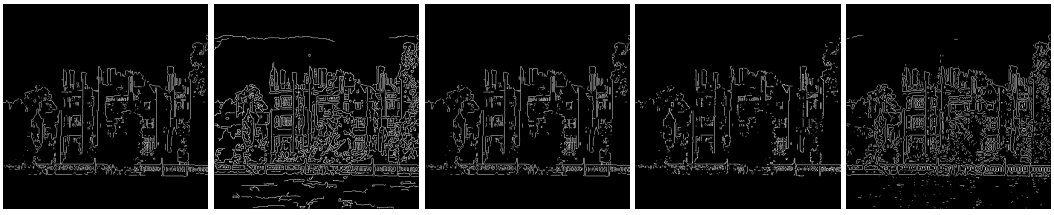
Figure D.3 Image aikftgxhesyxsvwx



(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System

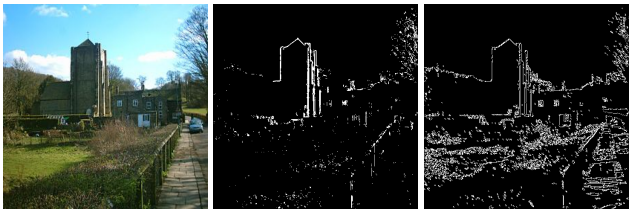


(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

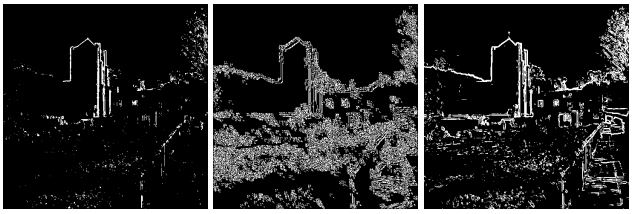


(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

Figure D.4 Image aipzlmztzfaesuqi



(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System

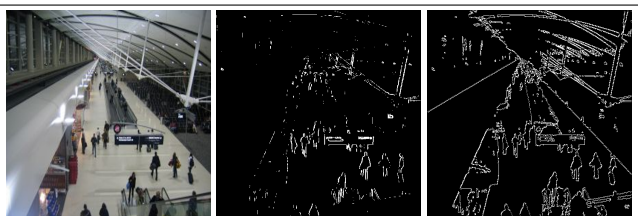


(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

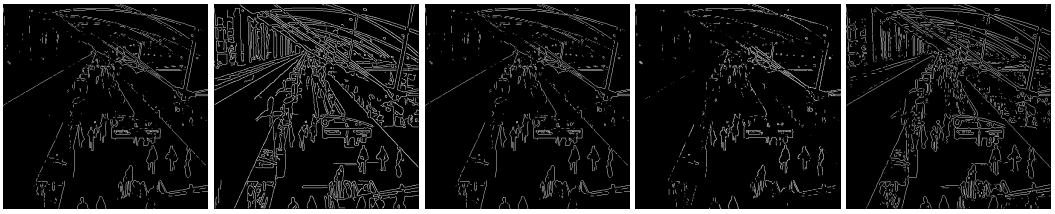
D.2 airport

Figure D.5 Image aajluizjalpcfwkf



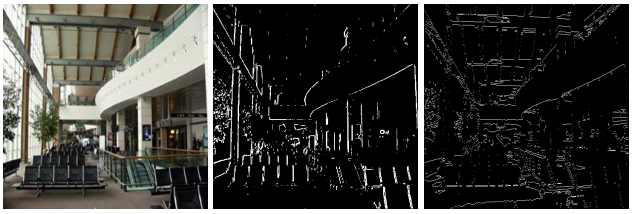
(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System

(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

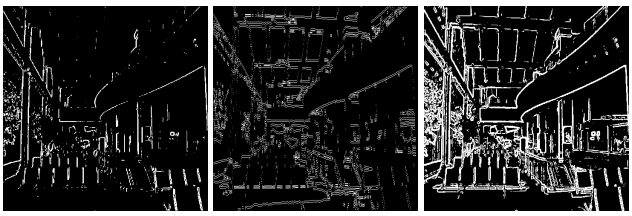


(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

Figure D.6 Image adqjafsbenxnqafr



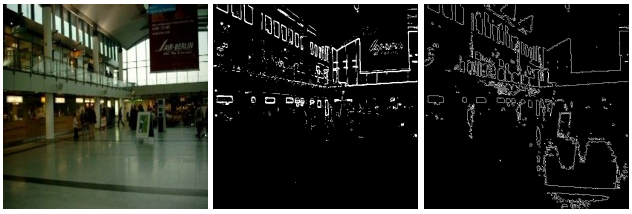
(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



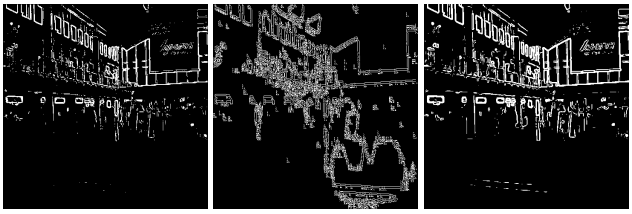
(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

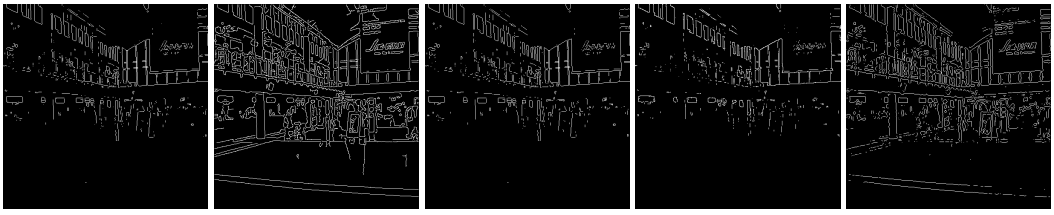
Figure D.7 Image adubahgmhbvxcuuy



(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



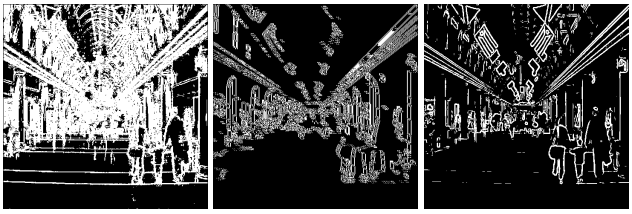
(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF



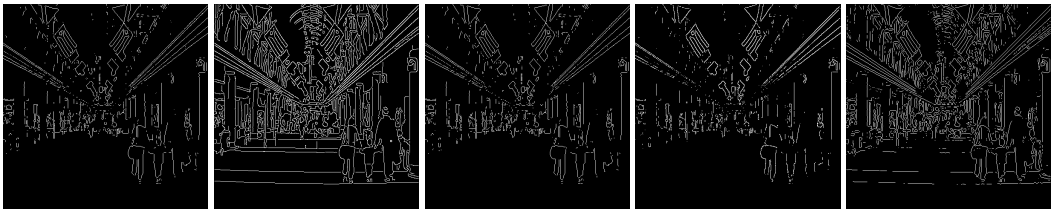
(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

Figure D.8 Image aesyuxjawitlduic

(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



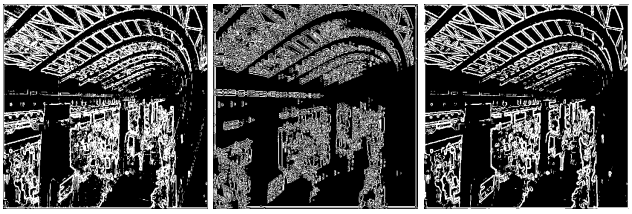
(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF



(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

Figure D.9 Image afqywdbbjtyksnnc

(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System

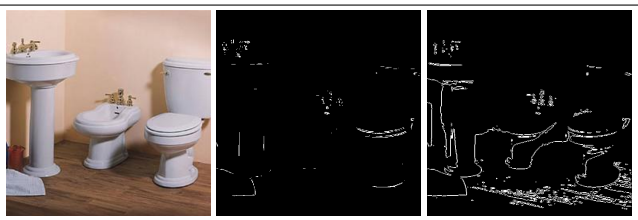


(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

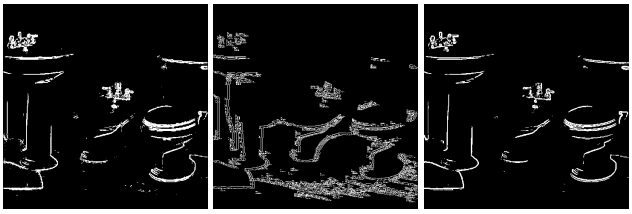
(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

D.3 bathroom

Figure D.10 Image aameczztdzpctgas



(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System

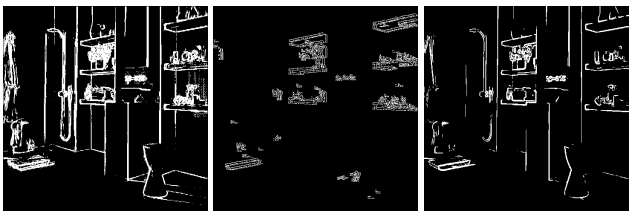


(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

Figure D.11 Image adcphrizvfpljiqt

(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



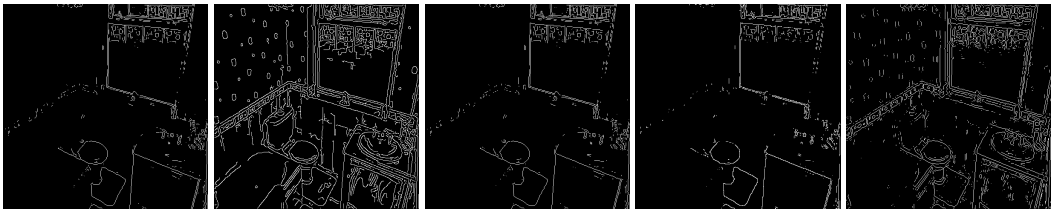
(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

Figure D.12 Image afpiqkoqucuomzrg

(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System

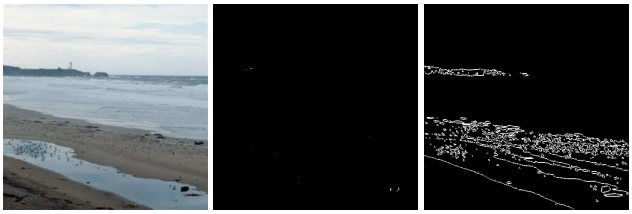
(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF



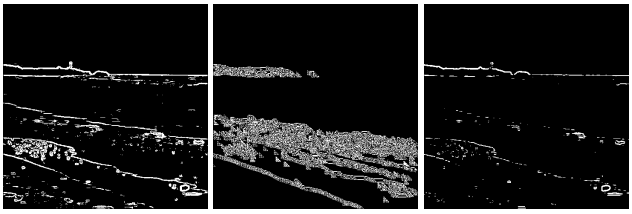
(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

D.4 beach

Figure D.13 Image ajijtsrervifddqo



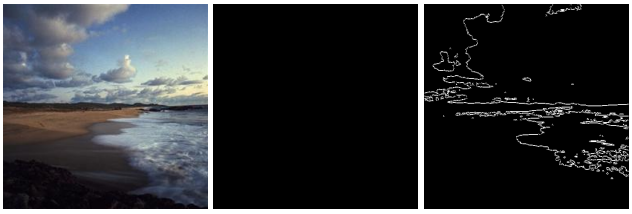
(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

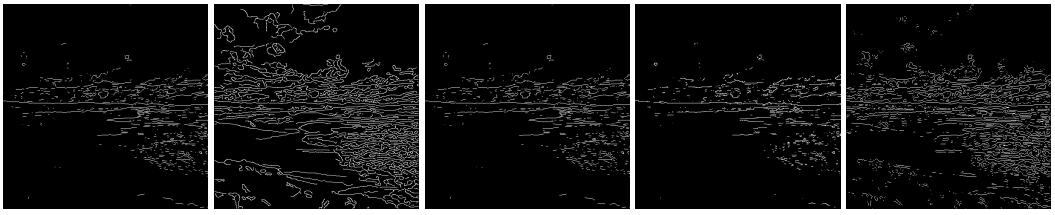
(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

Figure D.14 Image amjoyjrwhipnuvnm



(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System

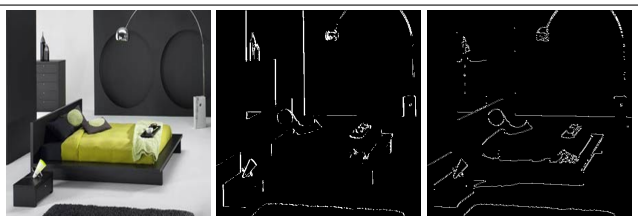
(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF



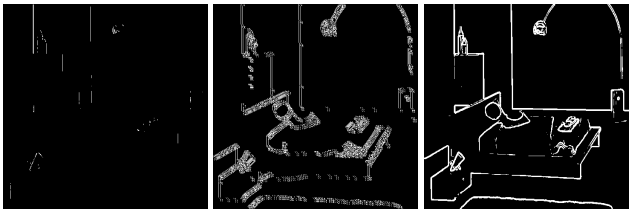
(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

D.5 bedroom

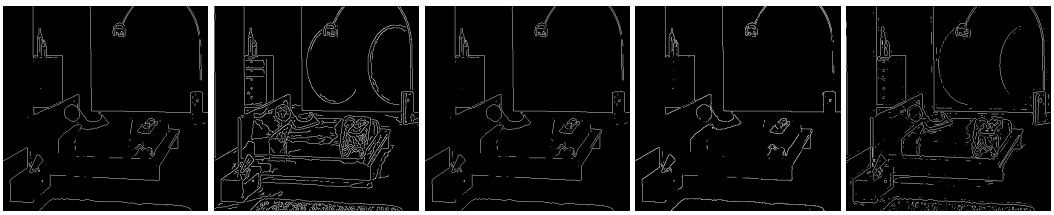
Figure D.15 Image aeknnscdzemmcqji



(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System

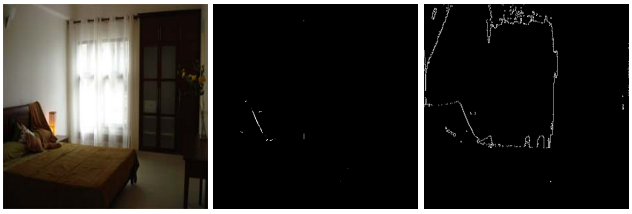


(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

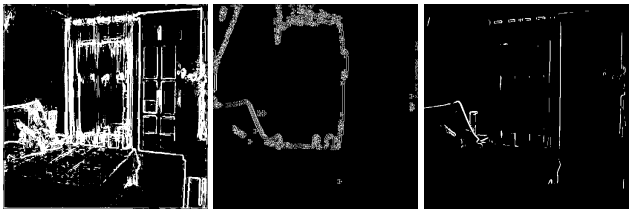


(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

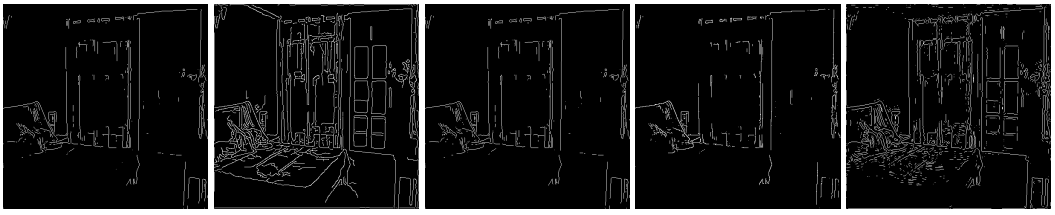
Figure D.16 Image aitlekhnfcgheoar



(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



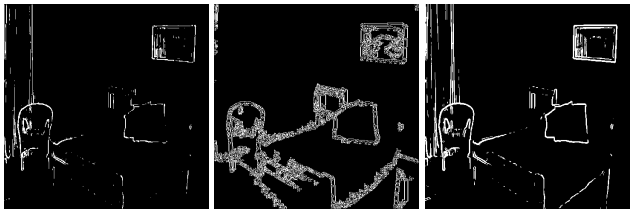
(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF



(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

Figure D.17 Image akpmyxoezscykphk

(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



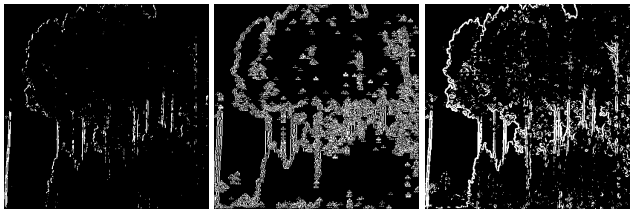
(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

D.6 broadleaf

Figure D.18 Image aaulapocjffusovo

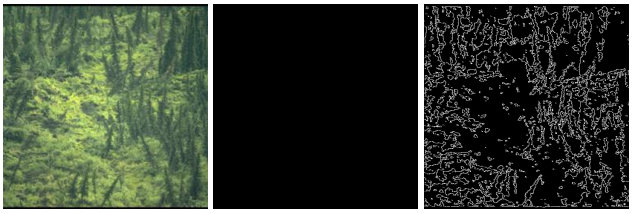
(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



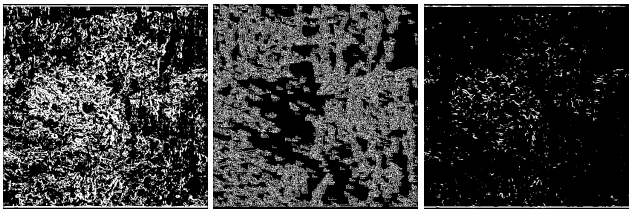
(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

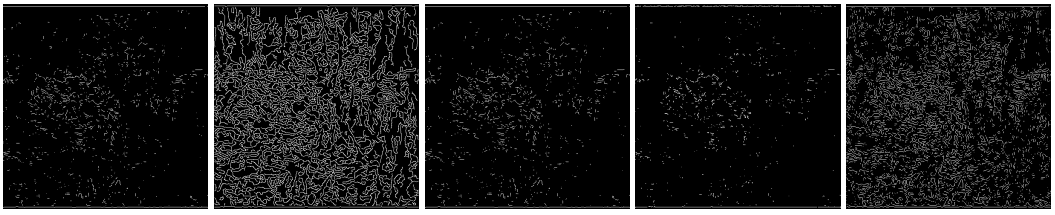
Figure D.19 Image aeswvtyqkhvasjui



(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



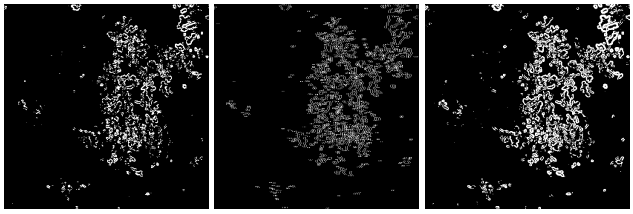
(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF



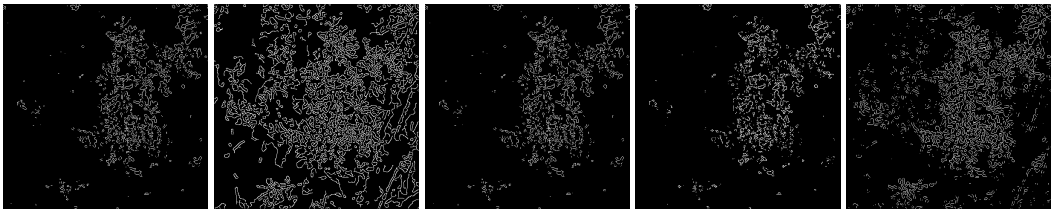
(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

Figure D.20 Image afqtsivgxsxbjlok

(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

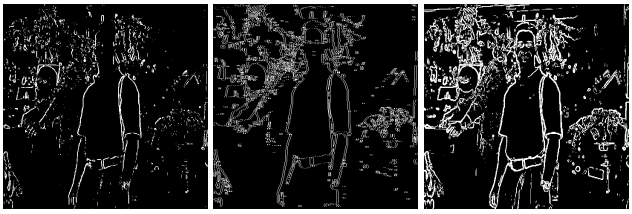


(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

D.7 candy store

Figure D.21 Image adomfndhzlfgqlvx

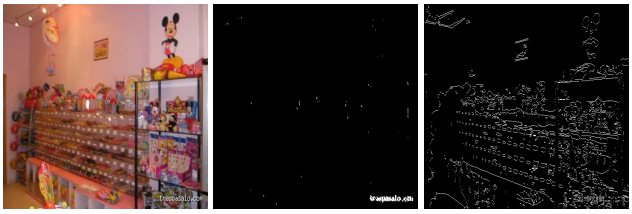
(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



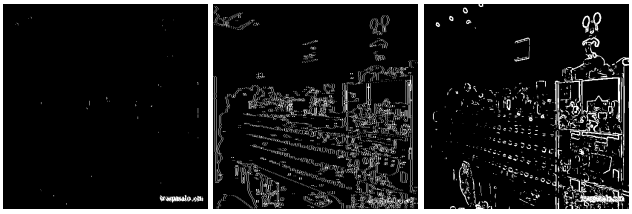
(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

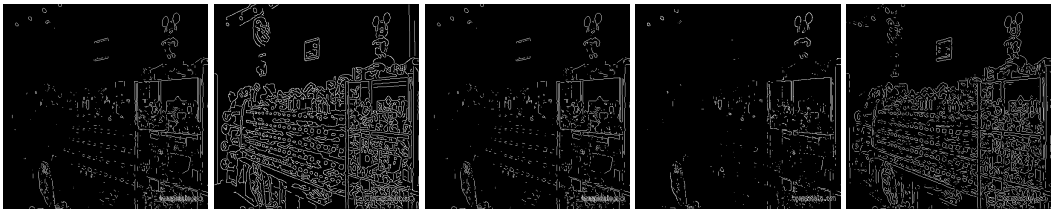
Figure D.22 Image amrzytvcreqnwqmq



(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System

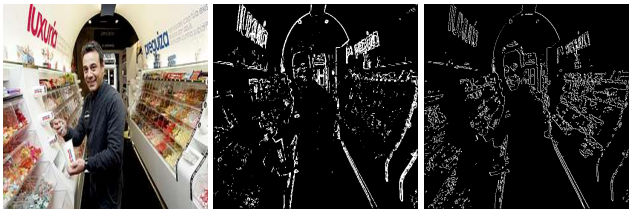


(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

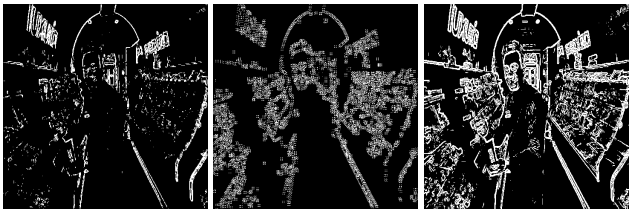


(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

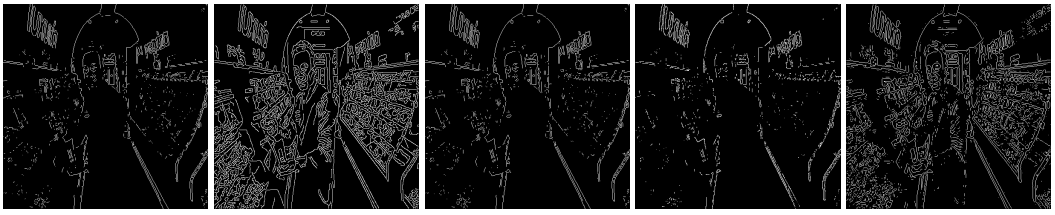
Figure D.23 Image apfiqqsakbuwpldm



(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



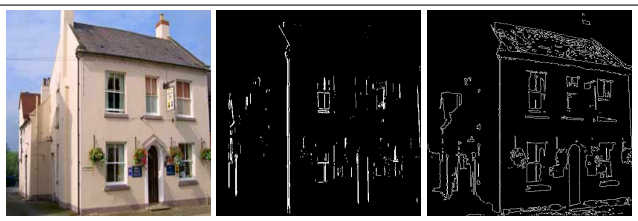
(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF



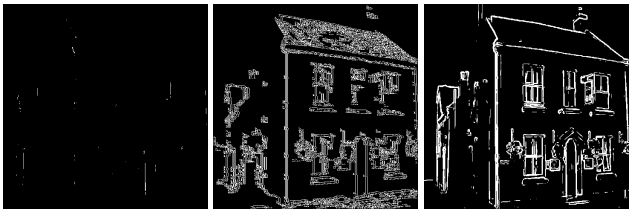
(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

D.8 house

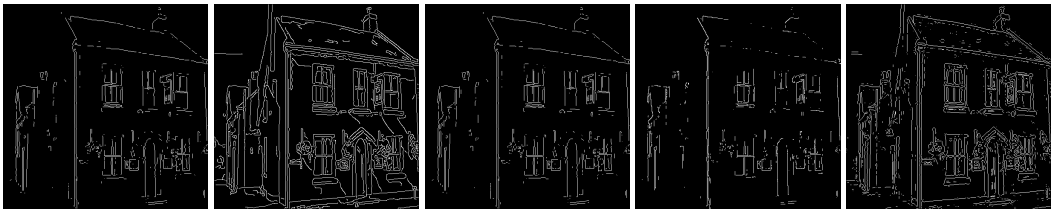
Figure D.24 Image akyeueuomjjjgzrn



(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF



(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

Figure D.25 Image amknzxaumkvwzfwh

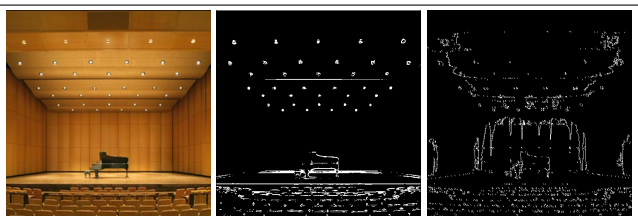
(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System

(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

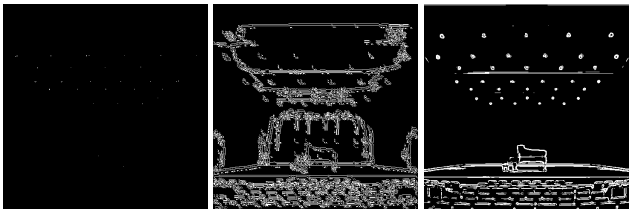
(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

D.9 indoor procenium

Figure D.26 Image alybzdyjzdhkxeiw



(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System

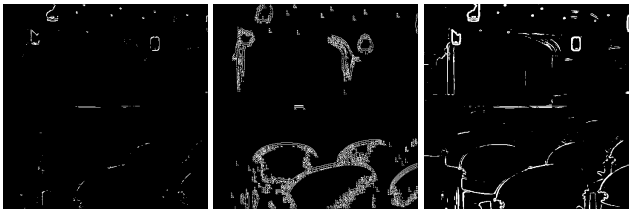


(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

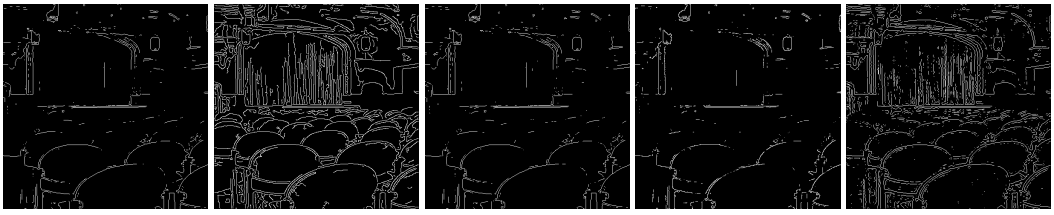
(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

Figure D.27 Image amkohsyogcolguzw

(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

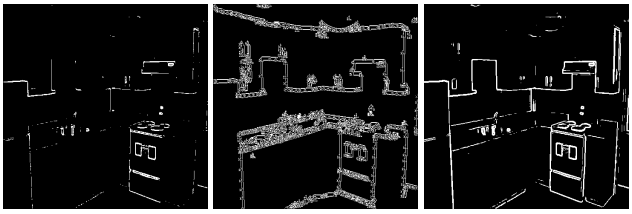


(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

D.10 kitchen

Figure D.28 Image actuxtkmvrrackkr

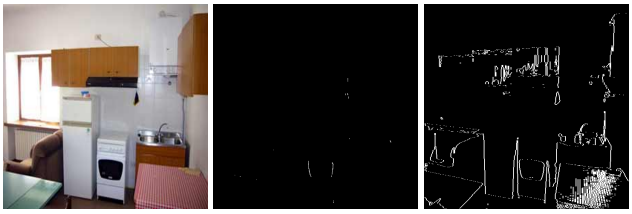
(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



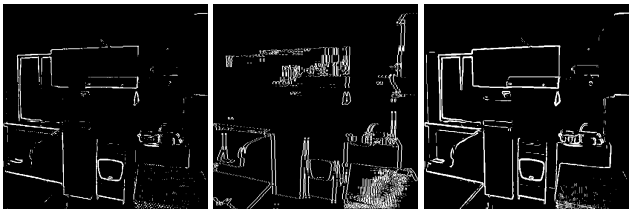
(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

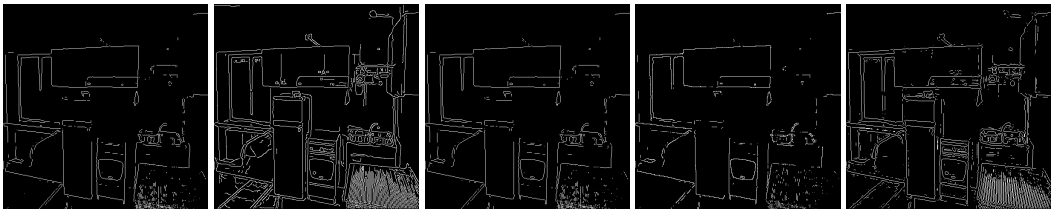
Figure D.29 Image atedeevhpbzyjtll



(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



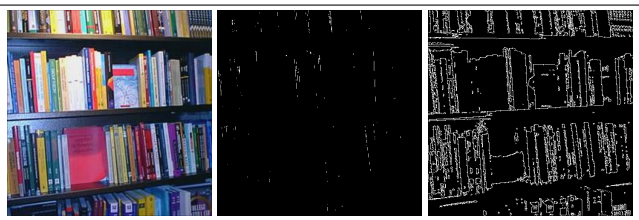
(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF



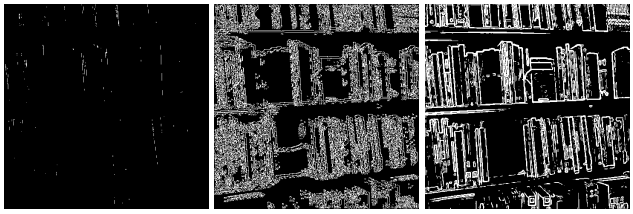
(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

D.11 library

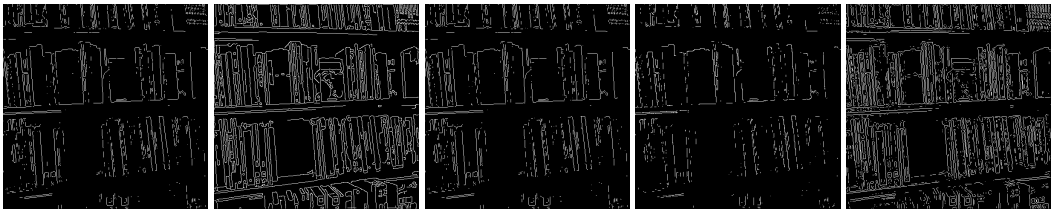
Figure D.30 Image ajqdbufkmatgfhkx



(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



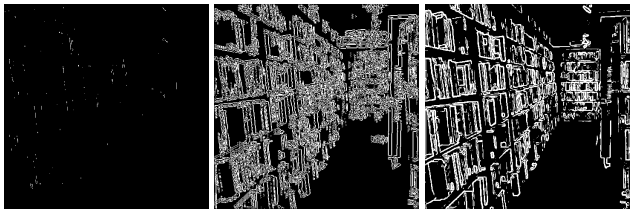
(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF



(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

Figure D.31 Image altzsxfweazxvpbz

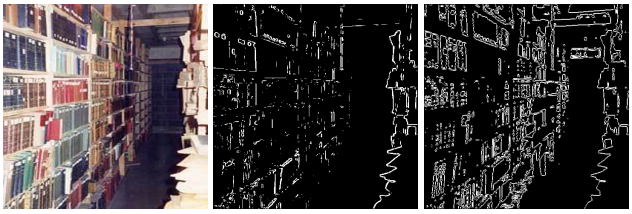
(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



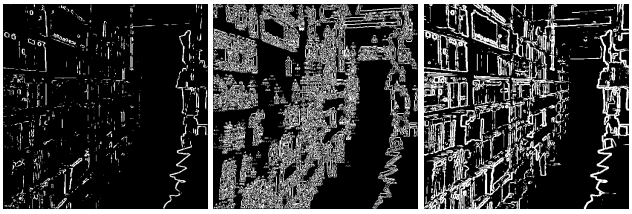
(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

Figure D.32 Image arcklkvfquuqjhgc



(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



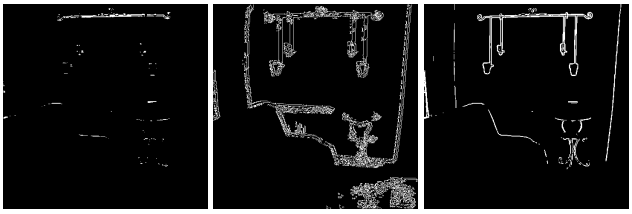
(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

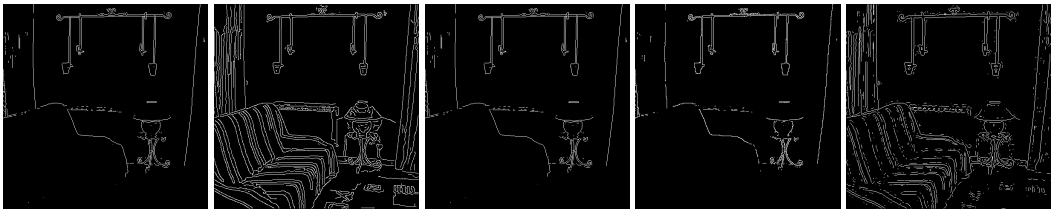
D.12 living room

Figure D.33 Image alizhxcsgtpjvrlz

(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



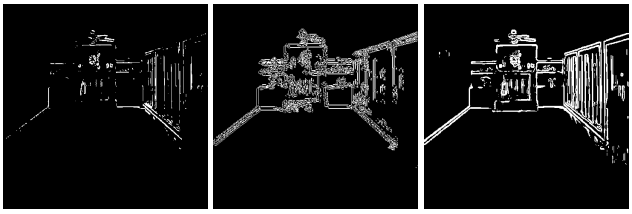
(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF



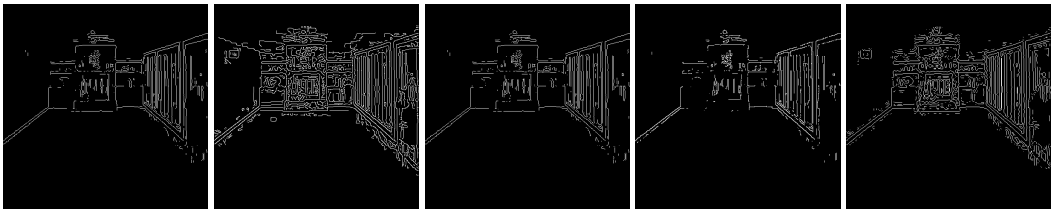
(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

Figure D.34 Image auxfadxjcaecurbm

(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



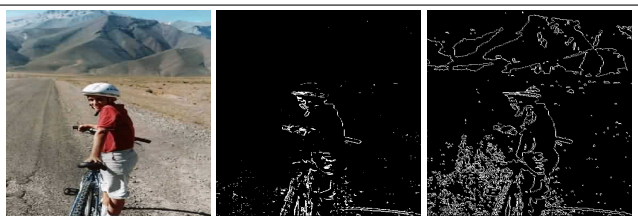
(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF



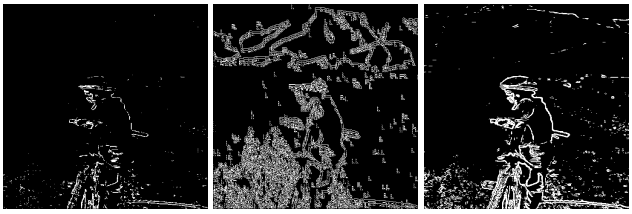
(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

D.13 moutain

Figure D.35 Image afliehgwaoiynwcw



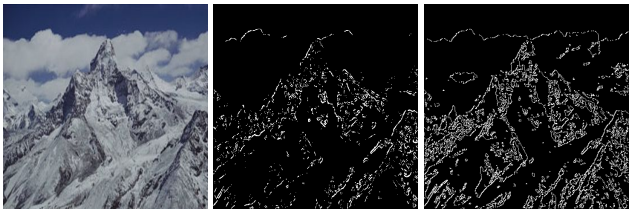
(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



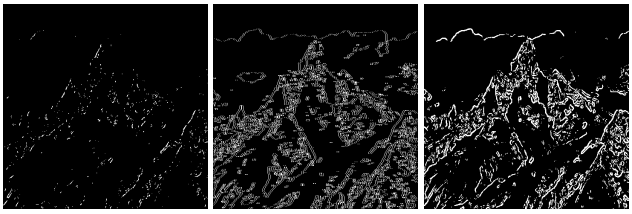
(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

Figure D.36 Image ajlvjxlreluvbtmn



(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



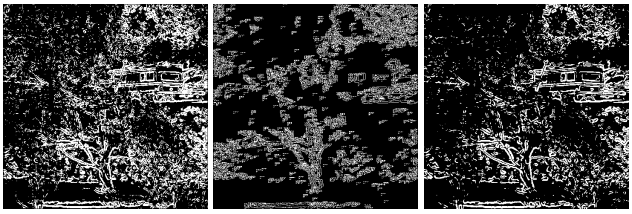
(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

D.14 river

Figure D.37 Image aanwybhpvpvtmrcr

(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System

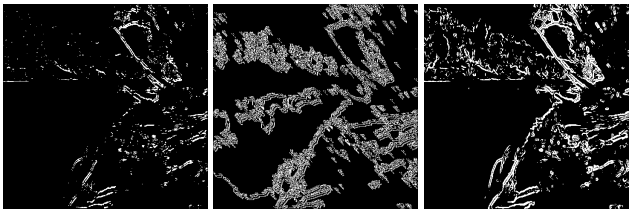


(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF

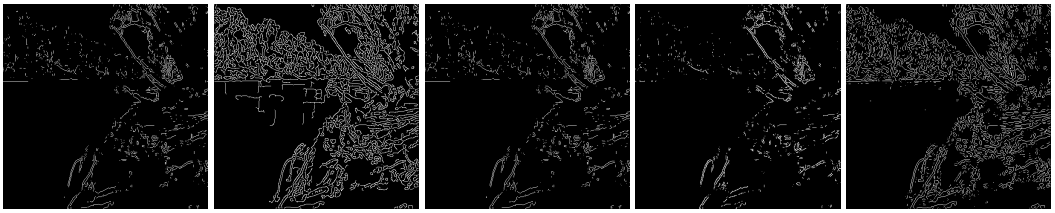
(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG

Figure D.38 Image abuxagxtyjqidams

(a) Original Image (b) Fuzzy System (c) Non-Fuzzy System



(d) Fuzzy System With Added Training Images 5 Times (e) Non-Fuzzy Filtered (f) CVF



(g) Sobel (h) Canny (i) Prewitt (j) Roberts (k) LoG